

N O T I C E

THIS DOCUMENT HAS BEEN REPRODUCED FROM
MICROFICHE. ALTHOUGH IT IS RECOGNIZED THAT
CERTAIN PORTIONS ARE ILLEGIBLE, IT IS BEING RELEASED
IN THE INTEREST OF MAKING AVAILABLE AS MUCH
INFORMATION AS POSSIBLE

**"Made available under NASA sponsorship
in the interest of early and wide dis-
semination of Earth Resources Survey
Program information and without liability
for any use made thereof."**

8.0 - 10199

JSC-12742

NASA Clk.

160671

DESIGN SPECIFICATION
FOR
LARSYS PROCEDURE 1

Job Order 71-695

(E80-10199) DESIGN SPECIFICATION FOR LARSYS
PROCEDURE 1 (Lockheed Electronics Co.)
145 p HC A07/MF A01 CSCL 05B

N80-28796

Unclas
G3/43 00199

Prepared By
Lockheed Electronics Company, Inc.
Systems and Services Division
Houston, Texas

Contract NAS 9-15200

For
EARTH OBSERVATIONS DIVISION
SCIENCE AND APPLICATIONS DIRECTORATE



National Aeronautics and Space Administration
LYNDON B. JOHNSON SPACE CENTER
Houston, Texas

April 1977

LEC-10417

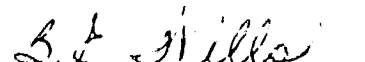


JSC-12742

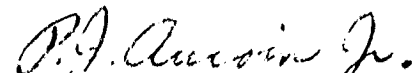
DESIGN SPECIFICATION
FOR
LARSYS PROCEDURE 1

Job Order 71-695

PREPARED BY


B. E. Wills



C. T. Gardner

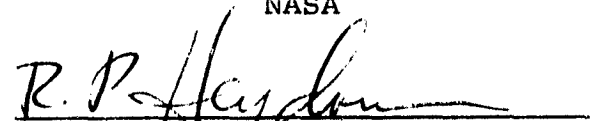

P. J. Aucoin, Jr.

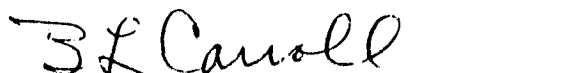
APPROVED BY

LEC

NASA


P. L. Krumm, Supervisor
Software Development Section


R. P. Heydorn
Procedure 1 Advisor
Research, Test, and Evaluation
Branch


B. L. Carroll, Manager
LACIE Development and
Evaluation Department

Prepared By

Lockheed Electronics Company, Inc.

For

Earth Observations Division
Science and Applications Directorate

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
LYNDON B. JOHNSON SPACE CENTER
HOUSTON, TEXAS

April 1977

LEC-10417

CONTENTS

Section	Page
1. SCOPE	1-1
2. APPLICABLE DOCUMENTS.	2-1
3. SYSTEM DESCRIPTION.	3-1
3.1 <u>HARDWARE DESCRIPTION</u>	3-2
3.2 <u>SOFTWARE DESCRIPTION</u>	3-4
3.2.1 SOFTWARE COMPONENT NO. 1 (DOTDATA)	3-4
3.2.2 SOFTWARE COMPONENT NO. 2 (LABEL)	3-12
3.2.3 SOFTWARE COMPONENT NO. 3 (ISOCLS)	3-26
3.2.4 SOFTWARE COMPONENT NO. 4 (SELECT)	3-32
3.2.5 SOFTWARE COMPONENT NO. 5 (CLASSIFY)	3-36
3.2.6 SOFTWARE COMPONENT NO. 6 (DISPLAY)	3-43
 Appendix	
A. SUBPROGRAMS FOR DOTDATA PROCESSOR	A-1
B. SUBPROGRAMS FOR LABEL PROCESSOR	B-1
C. SUBPROGRAM MODIFICATIONS FOR ISOCLS PROCESSOR	C-1
D. SUBPROGRAM MODIFICATIONS FOR SELECT PROCESSOR	D-1
E. SUBPROGRAM MODIFICATIONS FOR CLASSIFY PROCESSOR	E-1
F. SUBPROGRAM MODIFICATIONS FOR DISPLAY PROCESSOR.	F-1
G. DOT DATA FORMAT	G-1

1. SCOPE

This document contains the design specifications for implementing Procedure I in EOD-LARSYS. Software for two new processors will be written and routines for four existing processors will be modified.

EOD-LARSYS is operational on the Univac 1108 EXEC II version in Building 12. The system is batch oriented and operated and maintained according to IDSD procedures.

This document assumes the reader is familiar with both Procedure I and the LARSYS system.

2. APPLICABLE DOCUMENTS

- User Documentation EOD-LARSYS, Lockheed Electronics Company, Inc., HASD, Houston, Texas, November 1975
- Job Order: 63-1347-1695
- Requirements Document, Ref: 642-2240, dated February 7, 1977
- TIRF: 77-0008

3. SYSTEM DESCRIPTION

This document describes the LARSYS Procedure I. The section is intended to outline the sequence of executing the various processors to accomplish the task of classifying a LACIE segment.

The system is designed to provide RT&E with a means for experimenting with a new technique for classifying Landsat data for LACIE. The data will be pre-processed by ERIPS. ERIPS will merge tapes received from Goddard Space Flight Center and create a multi-temporal/multi-pass tape.

Using an ERIPS unload tape as input to the DOTDATA processor, a dot data file will be output. The file will contain both type 1 and type 2 dots. Type 1 dots are the dots to be used both as starting vectors for the clustering processor (ISOCLS) and as labeling vectors for the labeling processor (LABEL). Type 2 dots are the dots to be used as a bias correction factor in computing the classification results in the DISPLAY processor.

After delineating DO/DU fields by card input, ISOCLS will cluster the segment using the starting vectors from the dot data files to initial the clustering process. An unconditional cluster map and a set of 'unlabeled' stats will be output.

The 'unlabeled' stats, cluster map, and dot data file results will be input to the LABEL processor. Using one of two procedures, k-nearest neighbor or all-of-a-kind, the stats will be 'labeled'. A conditional or mixed cluster map may be output to be later displayed on the PMIS DAS or Image-100.

Using the 'labeled' stats, and allowing the user the capability of setting the inter-subclass weights by categories, the best k of n channels will be selected by the feature selection processor SELECT.

In CLASSIFY, each class in the 'labeled' stats may be assigned to a category by the analyst or by the system and the a priori for each category will be computed using the cluster population from the stats. Using the k best channels, the sum-of-density classifier will assign each pixel in the LACIE segment to a given subclass.

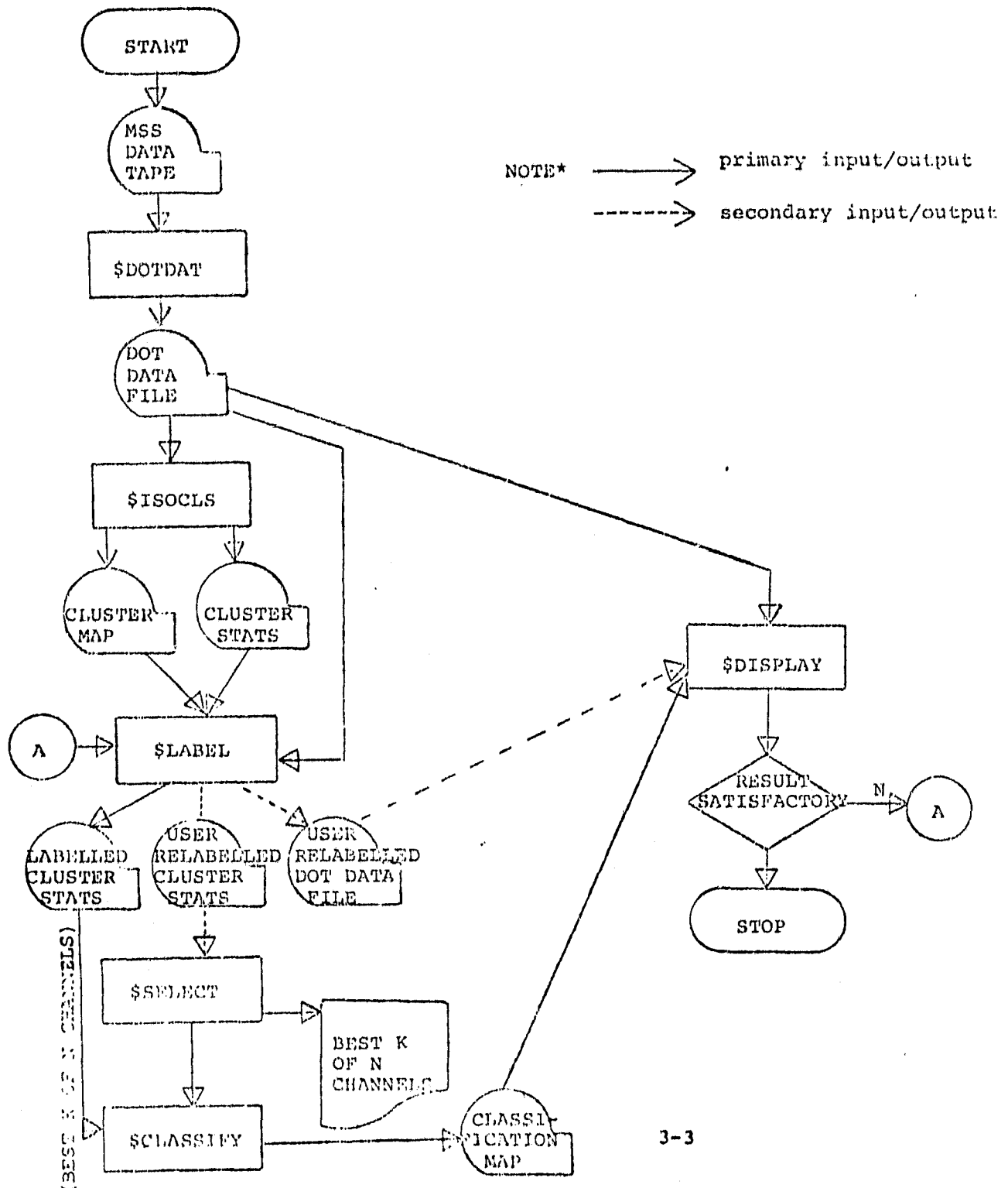
In DISPLAY, the bias correction dots will be used in computing the bias correction for the classified area. Two performance tables relating to the bias correction dots will be output. One table will contain a category dot summary performance; the other, an individual dot summary performance.

Given the analyst is not satisfied with the classification results, he may relabel the dots and/or relabel the stats (by a labeling procedure or by card input) and restart the process at any given point.

3.1 HARDWARE DESCRIPTION

N/A

PROCESSOR FLOW



3-3

ORIGINAL PAGE IS
OF POOR QUALITY

3.2 SOFTWARE DESCRIPTION

3.2.1 SOFTWARE COMPONENT NO. 1 (DOTDATA)

In implementing Procedure I, the means for allowing the user the capability to label certain MSS data points (known as dots) was added to the system by the processor DOTDATA. The main function of this processor is to output a file containing the dots of interest. This file will be an interface for three processors, ISOCLS, LABEL, and DISPLAY.

The dots are defined by field cards. By user control, any subset of the 209 possible grid points may be selected. Optionally, the dots may be labeled at the initial creation of the file or the dots may be labeled in the LABEL processor. All the categories of interest do not have to be defined in this processor.

If a dot is to be retyped, the file must be recreated by executing this processor again.

The file will contain both type 1 and type 2 dots. Type 1 dots (starting and labelling dots) will be written on one file; type 2 dots (bias correction dots), on a second file.

By an OPTION control card, the user may request that the spatial and spectral information relating to each dot on the file be printed on the line printer.

3.2.1.1 Linkages

The processor DOTDATA uses the FORTRAN-V compiler, Univac software system routines, EOD-LARSYS utility routines, and common blocks INFORM, GLOBAL and DOTVEC.

See Appendix A for description of subprograms.

3.2.1.2 Interfaces

N/A

3.2.1.3 Inputs

Requires an MSS data tape. See section 3.2 of LARSYS document.
Requires three types of card input.

- Processor Card

<u>Keyword</u>	<u>Function</u>
(Col. 1) \$DOTDATA	Loads into the system all the routines needed for executing this pro- cessor.

- New Control Cards

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(Col. 1)	(Col. 11-72)	
CHANNEL	DATA= n_1, n_2, \dots, n_{30} (Default: None)	Integer numbers separated by commas referring to the channels on the MSS data tape.
DATAFILE	UNIT= n , FILE= m (Default: $n=3$ $m=1$)	n is the FORTRAN unit number assigned to the MSS data tape. m is the file number of the data to process.
DOTFIL	INPUT/UNIT= n , FILE= m (Default: $n=8$ $m=1$)	n is the FORTRAN unit number assigned to the dot data file output by this processor, m is the number of the file to output.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
OPTION	PRINT (Default: no line printer output)	Initials the printing of the dot data file information.
HED1	Any 60 characters (Default: Lyndon B. Johnson Space Center)	First line of the heading on line printer output.
HED2	Any 60 characters (Default: Houston, Texas)	Second line of the heading on line printer output.
DATE	Any 12 characters (Default: Present date)	Date in the heading on line printer out- put.
COMMENT	Any 60 characters (Default: None)	Comment printed with heading on line printer output.
END		Indicates the end of the control card inputs.
\$END*		Indicates the end of all the card inputs.

● Field Cards

By the field card, the user will define the grid points to extract from the MSS data tape. The definition and order of the field card(s) will determine the position of the dots on the dot data file, DOTFIL. The analyst will need to know the position of the dots for defining starting vectors in ISOCLS and for labeling/relabeling the dots in LABEL.

At the time of defining the fields, the type for each dot will be defined by the TYPE card. By option, the analyst may label each dot by a CLASSNAME card. If this card is omitted, the unlabeled dots must be labeled by the control card DOTLABEL or excluded from the set by the control card EXCLUDE in the labeling processor, LABEL.

An example is given to illustrate a field data set expected by this processor.

```
*END*
TYPE      1
CLASSNAME  WHEAT (optional)
LABEL1     (10,10),(10,10),(196,10)
LABEL2     (10,10),(10,20),(196,20)
CLASSNAME  NONWHT (optional)
LABEL3     (10,10),(10,50),(100,50)
TYPE      2
CLASSNAME  WHEAT (optional)
BIAS1      (10,10),(10,40),(196,40)
CLASSNAME  NONWHT (optional)
BIAS2      (10,10),(10,70),(196,70)
$END*
```

Two files will be written. File 1 will contain 38 WHEAT dots, followed by 10 NONWHT dots. All of which are type 1 dots. File 2 will contain 19 WHEAT dots followed by 19 NONWHT dots. All of which are type 2 dots.

If the CLASSNAME cards were omitted, file 1 would contain 48 unlabeled, type 1 dots. File 2 would contain 19 unlabeled, type 2 dots.

In both cases, the reference number for the dots in file 1 defined by LABEL1 field card would be 1 thru 19; LABEL2 field

card would be 20 thru 38; and LABEL3 field card would be 39 thru 48. The reference number for the dots in file 2 defined by BIAS1 field card would be 1 thru 19; and BIAS2 field card would be 20 thru 38.

3.2.1.4 Outputs

DOTFIL, a multi-file unformatted FORTRAN written tape, is output as an interface to the processors ISOCLS, LABEL, and DISPLAY. The default unit for DOTFIL is logical unit F (FORTRAN unit 8). See Appendix G for format of the tape.

The line printer output will be:

- Summary of field information
- Spatial and spectral information for each dot (optional)
- Summary of user options

3.2.1.5 Storage Requirements

Requires less than 53K of core.

3.2.1.6 Description

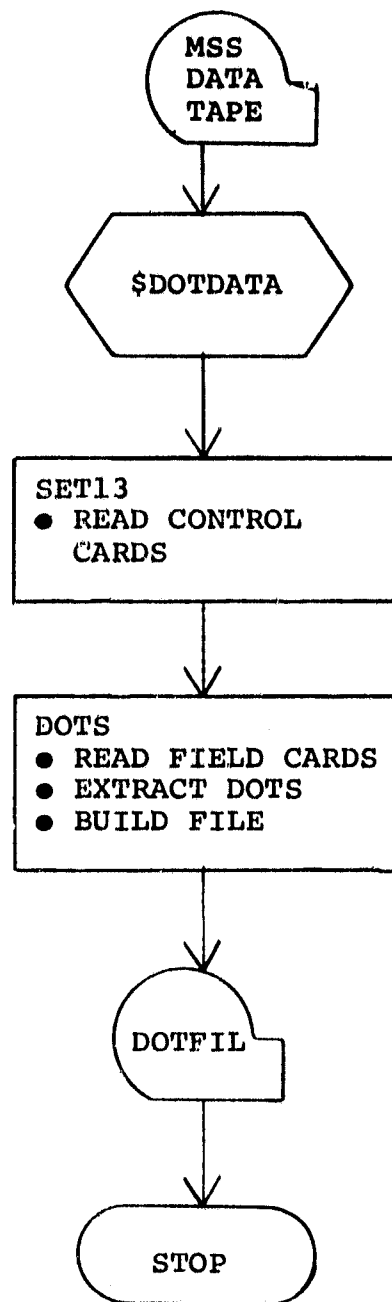
The main function of DOTDATA is to create the dot data file, DOTFIL. Optionally, the spatial and spectral information of each dot in the DOTFIL may be printed on the line printer.

DOTDAT will be the driver routine. From this routine, the setup routine SET13 will be called. SET13 will read and analyze the control cards and print a summary of the user selected options.

DOTS will be the coordinator and controller of the routines needed to create and output the file, DOTFIL.

From DOTS, FLDTYP is called to initiate the reading of the field cards. In returning from FLDTYP, the sample number, line number, type, category (optional), and data vector will be stored in the array DATA. DATA will be passed to the routine DOTFIL; this routine will output the dot data file.

3.2.1.7 Flowchart



3.2.1.8 Listing

TBD

3.2.2 SOFTWARE COMPONENT NO. 2 (LABEL)

To aid the analyst in supervising the labeling of the statistics obtained from the clustering processor ISOCLS, a new technique for labeling the statistics will be implemented.

Two procedures for labeling the statistics will be provided. Optionally, the analyst may select either the k- nearest neighbor procedure or the all-of-a-kind procedure.

Optionally, a conditional or mixed cluster map will be output to tape. Optionally, an unconditional cluster map will be output in the format acceptable by the DISPLAY processor.

The labels in the dot data file, DOTFIL and/or previously labeled statistics file, SAVTAP may be relabeled by control card input and the updated file output to tape.

3.2.2.1 Linkages

The processor LABEL uses the Fortran-V compiler, Univac software system routines, EOD-LARSYS utility routines, and common blocks INFORM, GLOBAL, and LABELER.

See appendix B for description of subprograms.

3.2.2.2 Interfaces

Optionally, the three files listed below may be input:

- Statistics file (SAVTAP) from ISOCLS or STAT processor
- Cluster map file (MAPFIL) from ISOCLS processor
- Dot data file (DOTFIL) from DOTDATA processor

For format of the files see section 4.2 and 5.1 in the LARSYS document and appendix G in this document.

3.2.2.3 Inputs

Requires 3 types of card input.

- Processor card

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(col. 1) \$LABEL		Loads into the system all the routines needed for executing this processor

- New control cards

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(col. 1)	(col. 11-72)	
CHANNEL	STAT= n_1, n_2, \dots, n_{30} DATA= m_1, m_2, \dots, m_{30} (Default: $n_i = \{\text{all channels on SAVTAP file}\}$ $m_i = \{\text{all channels on DOTFIL file}\}$	n_i and m_i are integer numbers separated by commas referring to the channels on the SAVTAP file and the MSS data tape, respectively.
DOTFILE	INPUT/FILE= m , UNIT= n	Defines the unit and files for the input dot data file DOTFIL. n is the Fortran unit number assigned to the input DOTFIL file. m is the number of the input file to process.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
DOTFIL	OUTPUT/FILE=m,UNIT=n (Default: NONE)	Defines the unit and file to which the re-labeled DOTFIL file is output. n is the Fortran unit number assigned to the output DOTFIL file. m is the number of the file to output.
STATFILE	INPUT/FILE=m,UNIT=n (Default: NONE)	Defines the unit and file for the input SAVTAP file. n is the Fortran unit number assigned to the input SAVTAP file. m is the number of the file to process.
STATFILE	OUTPUT/FILE=m,UNIT=n (Default: NONE)	Defines the unit and file to which the labeled/relabeled SAVTAP file is output. n is the Fortran unit number assigned to the output SAVTAP file. m is the number of the file to output.
MAPTAP	OUTPUT/FILE=m,UNIT=n (Default: no DISPLAY interface tape will be output.)	Defines the unit and file to which the unconditional cluster map, MAPTAP, is output. n is the Fortran unit number assigned to the output MAPTAP file.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
		If executing back to back with DISPLAY, n must be assigned to unit 2. m is the number of the file to output.
MAPFILE	INPUT/FILE=16,UNIT=n (Default: m=1)	Defines the unit and file for the input unconditional cluster map, MAPFIL, that is output by ISOCLS during the clustering process. n=16 is the Fortran unit number assigned to the input MAPFIL file. (n must be assigned to unit 16 if executing back to back with ISOCLS). m is the number of the file to process.
MAPFILE	OUTPUT/FILE=m,UNIT=n (Default: n=16 m=1)	Defines the unit and file to which the conditional or mixed cluster map is out-put. n is the Fortran unit number assigned to the output MAPFIL file. m is the number of the file to output.
DOTLABEL	category name, n ₁ ,n ₂ ,...,n ₂₅₀	The DOTFIL is labeled or relabel by this card. Category name is the label the analyst is assigning to the dots n _j , j=1,...,i. The category name may be

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
		composed of a maximum of 6 characters. n_i are integer numbers separated by commas referring to the position of the dot on the DOTFIL file.
STATLABEL	class name, n_i , n_2, \dots, n_{250} (Default: NONE)	The SAVTAP file may be manually relabeled by this card. n_j ($j=1, 2 \dots i$) are the number of the subclasses on the SAVTAP that are to be regrouped into another class. Class name is the name of the class to which the subclasses n_j are to be reassigned. Class name must match a name on the SAVTAP file.
DISTANCE	L1 (Default: L1 distance)	The L_1 distance will be used in computing the distance between the means of the cluster and the labeling dots.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
DISTANCE	L2 (Default: L ₁ distance)	The L ₂ distance will be used in computing the distance between the means of the cluster and the labeling dots.
OPTION	COND (Default: NONE)	A conditional clustered map will be output.
OPTION	MIXED (Default: NONE)	A mixed clustered map will be output.
THRESHOLD	T (Default: T=25.0)	T is the threshold parameter used in creating the conditional cluster map. T is a floating point number.
NEAREST	K (Default: K=1)	K is the number of dots to be used in the k-nearest neighbor procedure. K is an integer number. K ≤ 250.
PROCED	NAME (Default: N=K-NEAREST)	NAME is an alpha word. NAME=K-NEAREST (Use the k-nearest neighbor procedure.) NAME=ALL (Use the all-of-a-kind procedure.)

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
		NAME=MANUAL (Use the manual procedure of relabeling the DOTFIL or SAVTAP file.
MODULE	(blank)	Initiates the input of the module STAT card deck. The deck must immediately follow this card.
EXCLUDE	n_1, n_2, \dots, n_{250} (Default: all dots on the DOTFIL will be used.)	n_i are integer numbers referring to the dots on the DOTFIL that are to be excluded in all calculations (i.e., dots within a DO/DU area).
SUNANG	m_1, m_2, \dots, m_i (Default: no sun angle correction will be applied.)	m_j are integer sun angle numbers used in computing the L_1 or L_2 distances. A sun angle must be input for each acquisition of interest. An acquisition is assumed to be a 4 channel pass. Example: If the distance is computed using 16 channels, 4 sun angles (m_1, m_2, m_3, m_4) must be input.
SUNANG	FILE (Default: No sun angle correction will be applied.)	Sun angles will be extracted from the DOTFIL.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
HED1	Any 60 characters (Default: Lyndon B. Johnson Space Center)	First line of the heading on line printer output.
HED2	Any 60 characters (Default: Houston Texas)	Second line of the heading on line printer output.
DATE	Any 12 characters (Default: Present date)	Date in the heading on line printer output.
COMMENT	Any 60 characters (Default: NONE)	Comment printed with heading on line printer output.
END		Indicates the end of the control card inputs.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
\$END*		Indicates the end of all the card inputs.

- FIELD CARDS

If a procedure is selected, a MAPFIL tape must be input and a field card defining the area of the unconditional cluster map (input cluster map) must also be input. The vertices must reflect the sample number and line number of the MSS data tape used to create the unconditional cluster map. That is, the field card must be the identical to the field card input to ISOCLS.

If a procedure is not to be executed, a field card is not input.

Example of the data set:

- MAPFIL is being input

END

FIELD (1,1), (1,1), (196,1), (196,117), (1,117)

\$END*

- MAPFIL is not being input

END

\$END*

3.2.2.4 Outputs

Optionally the following files are output:

- Labeled statistic file (SAVTAP) by using a procedure (See section 4.1 of LARSYS document for format of file)

- Labeled statistic file (SAVTAP) by control card input
- Relabeled dot data file (DOTFIL) (See Appendix G for format of file)
- A conditional cluster map (See section 5.1 of LARSYS document for format of file)
- A mixed cluster map.
- An unconditional cluster map in the format acceptable by the DISPLAY processor (See Appendix D of LARSYS document for format of file)

A line printer summary of the following are output:

- Summary of selected options
- Summary of L_1 or L_2 distances
- Summary of the labeling dots within a cluster for the all-of-a-kind procedure
- Summary of the labeling dots for the k nearest dots to a cluster for the k-nearest neighbor procedure

3.2.2.5 Storage Requirements

Requires about 53k of core.

3.2.2.6 Description

A distance table computed using the L_1 or L_2 distance of the k-nearest labeling dots between the means of the cluster and the k dots will be computed.

$$L_1 = \sum_{i=1}^n |x_i - u_i|$$

$$L_2 = \sqrt{\sum_{i=1}^n (x_i - u_i)^2}$$

where:

x_i = i^{th} element of the dot vector

u_i = i^{th} element of the mean vector

n = number of channels

If applicable, a second distance table containing the distances between the mean of the cluster and the labeling dots within a cluster will be computed.

Using one or both tables as input to the k-nearest neighbor procedure or all-of-a-kind procedure, the statistics generated during clustering will be labeled.

For the k-nearest neighbor procedure, all the labels of the k-nearest labeling dots to a given cluster are polled. The label with the majority of the dots will label the clusters. If a tie occurs, then k-1 dots will be considered.

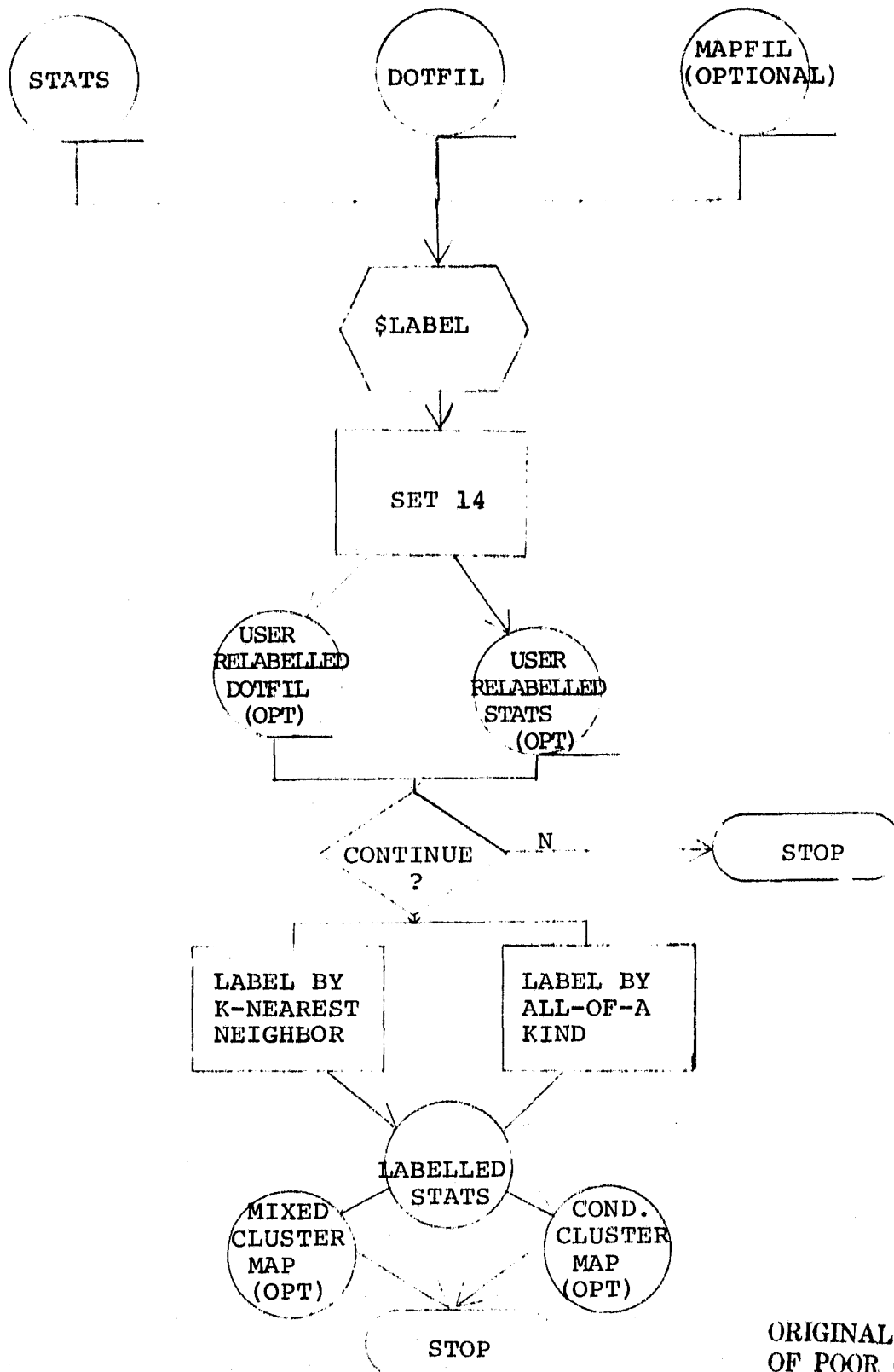
For the all-of-a-kind procedure, all of the labeling dots within a cluster are polled. If all the dots are of one category, the cluster will be labeled that category. If the cluster contains dots for more than one category, the label with the majority of the dots will label the cluster. If there are no labeling dots within a cluster, the labeling will default to the k-nearest neighbor procedure.

Optionally, a conditional cluster map may be output. A cluster is tagged as conditional if the distance between the nearest identically labelled labeling dot and mean of the cluster is greater than the analyst input threshold value t .

Optionally, a mixed cluster map may be output. A cluster is tagged as mixed if the labeling dots within a cluster are of more than one category.

Optionally, an unconditional cluster map may be output in the format acceptable to the DISPLAY processor. Information used in the thresholding procedure in DISPLAY will be dummed. If thresholding of the clustered data is desired, it can be performed by exercising the conditional map option in this processor.

3.2.2.7 FLOW CHART



ORIGINAL PAGE IS
OF POOR QUALITY

3.2.2.8 Listing

N/A

3.2.3 SOFTWARE COMPONENT NO. 3 (ISOCLS)

Several additions will be made to the ISOCLS (clustering) processor of the LARSYS system in support of Procedure 1 requirements. These additions are described in a general sense as follows:

- a. An option will be provided to use starting dots (pixels) from the dot data file DOTFIL to begin clustering.
- b. The analyst will be able to identify "designated other" and "designated unidentifiable" pixel sets (fields) by field card input. The pixels in these fields will not be included as inputs to the clustering algorithm. They will be assigned special cluster numbers and mean vectors for display purposes.
- c. An option will be provided to modify the pixel radiance values using a sun angle correction table. The correction table will be built in. The radiance value correction will apply only for clustering purposes.

The user will have the option of inputting the sun angles or requesting that these angles be extracted from the header record of a universal formatted MSS data tape (ERIPS unload tape).

- d. The following data will be extracted from the header record of a universal formatted MSS input tape:

Starting pixel number for the segment,
Stop pixel number,
Pixel skip factor,
Scan line skip factor.

3.2.3.1 Linkages

The processor ISOCLS uses the FORTRAN-V compiler, Univac

software system routines, EOD-LARSYS utility routines and common blocks PASS, GLOBAL and ISOLNK.

See Appendix C for description of subprograms modified.

3.2.3.2 Interfaces

Files generated by other processors and used as input to ISOCLS are:

- a. On option, a statistics file (SAVTAP) to provide starting cluster mean vectors
- b. On option, input initial cluster "centers" by cards
- c. On option, starting dots from DOTFIL to initialize the cluster processing

Format descriptions of these files are included in the LARSYS documentation in sections 4.1 and 3.1.4.3, and Appendix G in this document, respectively.

3.2.3.3 Inputs

Requires a MSS data tape (see section 3.2 of LARSYS document for format).

Requires additional control cards.

● New Control Cards

<u>Keyword</u>	<u>Parameters</u>	<u>Function</u>
DOTFILE	INPUT/UNIT=n, FILE=m (Default: Self-initializing starting)	Defines the FORTRAN unit number n and file number m of the dot data file DOTFIL containing the starting vectors.
DOTS	n_1, n_2, \dots, n_{60} (Default: Dots will not be used for starting vectors.)	n_i are integer numbers separated by a comma specifying the dots to be used as starting vectors.

<u>Keyword</u>	<u>Parameters</u>	<u>Function</u>
SUNANG	TAPE (Default: No sun angle correction applied)	Sun angles will be extracted from the ERIPS unload MSS data tape.
SUNANG	n_1, n_2, \dots, n_j n_j are integer numbers, $j \leq 7$. (Default: No sun angle correction applied)	$\{n_j\}$ are the sun angles to be used in computing the sun angle corrections for use in the cluster- ing algorithm. A sun angle must be input for each set of 4 channels input on the CHANNEL control card.

- Field Cards

ISOCLS will be modified to recognize DO/DU fields. All the DO/DU field cards (for all classes) must be input before the fields to be clustered. These fields must immediately follow the *END* card. The CLASSNAME card will follow the last DO/DU field card.

Example:

- If DO/DU fields are being defined

END

DESIGNATED OTHER

OTHER (1,1), (1,1), (40,1), (40,20), (1,20)

DESIGNATED UNIDENTIFIABLE

UNIDEN (1,1), (5,7), (8,7), (8,10), (5,10)

CLASSNAME WHEAT

WHT1 (1,1), (1,1), (196,1), (196,117), (1,117)

\$END*

- If no DO/DU fields are being defined

END

CLASSNAME WHEAT

WH1 (1,1), (1,1), (196,1), (196,117), (1,117)

\$END*

3.2.3.4 Outputs

- Statistics file (SAVTAP) and on option, punched card deck
- Cluster map tape (MAPFIL), on option
- Printout of cluster results, consisting of the following data items by class:

Cluster numbers and symbols

Cluster mean vectors (by channel)

Cluster standard deviations by channel

Inter-cluster distances

Number of pixels per cluster

Number of clusters

Cluster map by field for each class

3.2.3.5 Storage Requirements

TBD

3.2.3.6 Description

The implementation of the additions to ISOCLS will be generally described. For further details, please refer to the subprogram documentation which is included.

- By expanding the INVEC data vector of the control card reader/interpreter SETUP7, a DOTFILE control card will be accepted. Specification of unit and file number on this card will trigger a call to routine RDDOTS to load

the starting vectors. These vectors will be stored internally for use by ISODAT. The unit and file number will be stored in an extension to labeled common block GLOBAL.

- b. Pixel fields to be described as "other" or "unidentifiable" will be input by analyst-furnished classname and field definition cards to be read by RDDATA. The classname cards will be punched as:

DESIGNATE OTHER or
DESIGNATE UNIDENTIFIABLE.

The DO/DU pixels will be flagged and assigned special cluster numbers. They will not be included as input to the clustering algorithm. This requires a modification to RDDATA and ISODAT.

Field definition information will be stored internally for use by ISODAT. A trigger indicating that DO/DU fields have been used will be set and stored in an extension of the labeled common block PASS.

- c. To handle sun angle input, SETUP7 will be modified (by adding 'SUNANG' to the INVEC data vector) to read a sun angle control card. This card will either contain in the parameter field of the card the sun angles for each pass or the word TAPE. TAPE triggers the unpacking of sun angles from the MSS data tape. If the SUNANG control card is not present, a default to no sun angle correction will occur. The trigger indicating that the sun angles have been unpacked will appear in labeled common ISOLNK.

The TAPHDR section of subprogram TAPERD will be modified to unpack the sun angles from the tape header record.

The sun angles will be stored in ISOLNK. The sun angle correction table will probably be stored as DATA in subprogram ISODAT, although this is subject to change if a decision is made to do this correction in other processors. The correction is done by looking up the gain vs sun angle for each channel and multiplying the pixel radiance value for the channel by the gain.

- d. The pixel start number, stop number, skip factor and line skip factor values will be extracted from the header record of a universal format MSS data tape by a modification to TAPHDR. (See item 3 above.) These values will be stored in the ISOLNK labeled common.

3.2.3.7 Flowchart

N/A

3.2.3.8 Listing

N/A

3.2.4 SOFTWARE COMPONENT NO. 4 (SELECT)

The SELECT processor will be modified to optionally provide an automatic assignment of interclass weights. The weights for $class_i$, $class_j = 1.0$ for $i \neq j$ and the weights for $class_i$, $class_j = 0.0$ for $i = j$. The breakdown of $class_i$, $class_j$ pairs into the correct set of intersubclass pairs will be provided by the processor. Information concerning the class-subclass association will be extracted from the input statistics file, SAVTAP.

The modifications to the SELECT processor involve two subprograms, SETUP4 and WGTCHK. These modifications will be made to enable use of the processor in the evaluation of the LACIE Procedure 1 crop classification technique. For application of the SELECT processor to LACIE Procedure I, the user will consider classes equivalent to categories.

3.2.4.1 Linkages

The SELECT processor uses the Univac 1108 (EXEC2) operating system and associated system routines, the FORTRAN V compiler, EOD-LARSYS utility routines, and EOD-LARSYS common blocks FSL, GLOBAL, and INFORM.

See Appendix D for the detailed description of modifications to the SETUP4 and WGTCHK subprograms of the SELECT processor.

3.2.4.2 Interfaces

The SELECT processor interfaces with the LABEL processor via the labeled statistics file, SAVTAP, which is output by the LABEL processor. The statistics file is also the interface between SELECT and the STAT or ISOCLS processors.

3.2.4.3 Inputs

The modifications to the SELECT processor will include an additional input option on the currently available OPTION control card. The new input option will be required to initiate the automatic (processor provided) assignment of interclass weights. See below for the format and function of the new option.

- Revised Control Card

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(col. 1) OPTION	(col. 11-72) CLSWT (Default: The weights are assigned to intersubclass pairs.)	The processor determines the class-subclass correspondence (after any grouping of subclasses to form new subclasses if the GROUP control card is used) and assigns a weight=1.0 to the subclass pairs associated with all interclass pairs. Intraclass subclass pairs are given a weight=0.0

NOTE(1): The WEIGHTS control card remains available to allow the user to set weights for specific subclass pairs. If used, the input subclass pair weights override the processor-set subclass pair weights.

NOTE(2): The "WEIGHTS OTHERS" capability is not available when this option is exercised. If input, it is ignored by the processor.

3.2.4.4 Outputs

Line printer output and file output will be unchanged by these modifications to the SELECT processor.

3.2.4.5 Storage Requirements

TBD

3.2.4.6 Description

When the new option is input via the OPTION CLSWT control card, the SELECT processor sets an internal flag indicating that subclass pair weights are to be determined by the processor on an interclass bases. The processor determines the subclass-to-class association using the class (or category) and subclass information available from the input statistics file, SAVTAP. When determining the class-subclass association, the processor recognizes and uses any new subclasses created by the user via the currently available GROUP control card. Using storage locations already available, the processor builds a table of subclass pair wights for all subclass pairs involved in interclass pairing. Each and every subclass pair which represents an interclass pair is given a weight=1.0. All other subclass pairs (i.e., intraclass subclass pairs) are given a weight=0.0.

The process will continue to honor any subclass pair weights input by the user via the currently available WEIGHTS control card. Any input subclass pair weights will be placed in the weight table overriding the processor-set weights for those subclass pairs.

The processor will ignore the currently available WEIGHTS OTHERS control card if input along with the OPTION CLSWT control card.

3.2.4.7 Flowchart

N/A

3.2.4.8 Listing

TBD

3.2.5 SOFTWARE COMPONENT NO. 5 (CLASSIFY)

The classification processor, CLASSIFY, will be modified as follows:

1. To allow an option for obtaining subclass a priori values using subclass population data from the input file, SAVTAP.
2. To allow the system to assign the category names using the class names from the input statistics file, SAVTAP.

Both options are an addition to the current capability of analyst-input of a priori probability values at the subclass, class, or category level via the APRIORI control card and of category name input via the CATEGORY control card.

3.2.5.1 Linkages

The CLASSIFY processor uses the Univac 1108-EXEC 2 operating system and associated system routines, the FORTRAN V compiler, and EOD-LARSYS utility routines and common blocks. See Appendix E for the description of subprogram(s) modified to incorporate the additional processor capability described in Section 3.2.5

3.2.5.2 Interfaces

The CLASSIFY processor obtains the class names, subclass statistics (mean vectors and covariance matrices) and related subclass data from either an input statistics file, SAVTAP, or the Module STAT card deck which are created by either the STAT processor or the ISOCLS processor. The CLASSIFY processor also interfaces with the SELECT processor to obtain a transformation matrix via either a file, BMFILE, or a B-MATRIX card deck which are created by the SELECT processor.

The input statistics file, SAVTAP, is either assigned to UNIVAC unit A or to the users choice of available units. The input transformation matrix file, BMFILE, is always assigned to UNIVAC unit H.

3.2.5.3 Inputs

- MSS Data

The MSS (multi-spectral scanner) data to be classified is input to the CLASSIFY processor via the input file, DATAPE. The file must be in either of two specific formats, LARSYS II or Universal, and is either assigned to Univac unit C, or to the user's choice of units as designated on the DATAFILE control card.

- REVISED PROCESSOR CONTROL CARD

Changes to the control card input for the CLASSIFY processor to initiate the modification described in Section 3.2.5 are on the APRIORI control card and CATEGORY control card.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(col. 1)	(col. 11-72)	
CATEGORY	FILE (Default: No categories are defined and the standard classifier will be applied)	Initials the assigning of the category names using the class names from the input statistics file, SAVTAP, and invokes the category classifier.

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
APRIORI	<p>FILE</p> <p>(Default: Subclass a priori will not be computed from the statistics file, SAVTAP.</p>	<p>The subclass apriori probability values are computed using subclass or cluster point populations from the statistics file, SAVTAP.</p>
APRIORI	<p>A_1, A_2, \dots, A_m</p> <p>or</p> <p>$N * A_1, K * A_2, A_3, \dots$</p> <p>where N and K are arbitrary integer repetition factors, A_i are decimal numbers such that</p> $\sum_{i=1}^M A_i = 1.0$ <p>M = {</p> <ul style="list-style-type: none"> No. of subclasses or No. of classes or No of categories <p>(Default: If executing the standard classifier, each subclass is given equal apriori probability values. If executing the category classifier, each category is given equal apriori</p>	<p>The parameter field is expected to contain decimal numbers separated by a comma ",", which are the apriori probability values for either all subclasses, classes, or categories defined to the CLASSIFY processor. If the apriori values are input by class or category, the values will be distributed among the subclasses in the following manner:</p> <p>By class -</p> <p>$\text{Subclass}_i \text{Apriori} = \text{class Apriori} / \text{No. of subclasses in the class}$</p> <p>By category -</p> <p>$\text{Subclass}_i \text{Apriori} = \text{category Apriori} / \text{No. of subclasses in the category}$</p>

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(Continued)	probability values, and the apriori probability value for each category is subdivided equally among the subclasses in that category).	The order in which the apriori probability values, A_i , are input must be in the order in which the category, class or subclass is defined to the CLASSIFY processor.

- Field definition cards

The area(s) on the input data tape to be classified by the CLASSIFY processor are defined to the processor on input "field definition" card(s). The area(s) to be classified are defined in terms of sample, line coordinates of each vertex of the field, up to a maximum of ten (10) vertices for an irregularly shaped field. Also contained on the field definition card(s) are the incrementation of samples and/or lines to be performed in reading the data for the defined field.

The format of the "field definition" card is provided in Section 3.1.3 of the "EOD-LARSYS USER DOCUMENTATION", document no. LEC-3984.

3.2.5.4 Output

The classification processor, CLASSIFY, outputs the classification results on a file, MAPTAP, assigned to Univac Unit "B". The primary results output on the MAPTAP file are the subclass identification of each pixel of the area classified, and the value of the probability density function for each pixel of the area classified.

The classification processor outputs via the line printer a classification map, which illustrates the classified identity of each pixel by symbols, in correct spatial relationship according to sample, line coordinates of each pixel. Also output to the line printer are the class/subclass training field coordinates, the class/subclass statistics (mean vectors and covariance matrices) and a listing of the subclasses considered for classification along with apriori probability values for each subclass.

3.2.5.5 Storage Used

TBD

3.2.5.6 Description

To accomplish the additional option described in Section 3.2.5, the modifications to the CLASSIFY processor will involve two subprograms, REDIF2 and SETUP2.

Subprogram REDIF2 will be modified to read and decode the revised APRIORI control card and CATEGORY control card described in Section 3.2.5.3. If the word "FILE" is detected in the parameter field (card columns 11 through 72) of the APRIORI control card, an internal flag will be set to indicate that the subclass population data from the statistics file, SAVTAP, is to be read and used for computing subclass apriori probability values. If the word "FILE" is not detected in the parameter field of the input APRIORI card, it will be assumed that the card contains floating point (decimal) numbers, and the parameter field of the card will be read as is currently done - i.e., scanned for decimal numbers separated by a comma, and storing each number found into the apriori probability value array.

If the flag is set by virtue of detecting the word "FILE" in the parameter field of the APRIORI control card, subprogram SETUP2 will read the subclass population data from the statistics file, SAVTAP. The computation of subclass apriori probability values will be performed in SETUP2 and the computed values will be stored in the apriori probability value array. The computation of the subclass apriori probability values will be performed as follows:

$$P_i = \frac{N_i}{K}$$

where

P_i = apriori probability for subclass i

N_i = number of pixels in subclass i

K = total number of pixels in all subclasses

If the word 'FILE' is detected in the parameter field of the CATEGORY control card, each class defined on the SAVTAP file will define a category. For each class name there will be a corresponding category of like name.

If the word 'FILE' is not detected in the parameter field, it will be assumed that a category is being defined by analyst input.

In the subprogram SETUP2 three arrays will be initialized as follows:

KCLSNA = class names from SAVTAP file

NOCTCL = 1 for number of classes in category I

CATNAM = class names from SAVTAP file

and NOCAT = NOCLS2 where NOCAT is the number of categories being defined and NOCLS2 is the number of class names on the SAVTAP file.

3.2.5.7 Flow Chart

N/A

3.2.5.8 Listing

TBD

3.2.6 SOFTWARE COMPONENT NO. 6 (DISPLAY)

The classification results display processor, DISPLAY, will be modified as follows:

1. to accept the dot data file, DOTFIL
2. to provide a dot data classification performance summary by categories which includes:
 - a tabulation of both the uncorrected proportion and the bias correction proportion of each dot data category in the total area classified
 - an "alpha" table which tabulates the proportions of the bias correction dots (type 2 dots) on the basis of the analyst's labels versus the category classifier's labels of the bias correction dots
3. to provide a dot data classification performance summary of each dot on the analyst's specified file

3.2.6.1 Linkages

The DISPLAY processor requires the Univac EXEC2 operating system, the Fortran V compiler, the EOD-LARSYS utility routines, and common blocks GLOBAL and DISPL.

See Appendix F for a detailed description of modification to DISPLAY processor subprograms.

3.2.6.2 Interfaces

Optionally, the modified DISPLAY processor accepts as input the dot data file, DOTFIL, created by the DOTDATA processor. The file will be assigned either to Univac logical unit F (Fortran unit 8) or to an analyst specified unit. (See appendix G for format of file)

The DISPLAY processor requires the input of a classification results file, MAPTAP, created by the CLASSIFY processor. The file must be assigned to Univac logical unit B (Fortran unit 2). (See Appendix D in LARSYS User's document for format of file)

3.2.6.3 Inputs

Requires an additional control card.

● NEW CONTROL CARD

<u>Keyword</u>	<u>Parameter(s)</u>	<u>Function</u>
(col. 1)	(col. 11-72)	
DOTFILE	UNIT=n, FILE=m (Default: UNIT=8, FILE=1 No bias correction performance tables.)	Initiates the input of dot data file, DOTFIL, from the designated (or default, if not designated) unit and/or file, and initiates the output of the dot data classification performance summaries. The parameter, 'm', designates the file to be processed by DISPLAY. 'n' designates the Fortran unit number assigned to the input file, DOTFIL.

● FIELD CARDS

If the Procedure I option is to be exercised, the only kinds of fields that may be input are DO/DU fields (designated other or designated unidentifiable)

No test fields may be input; the training fields are the bias correction dots on the dot data file, DOTFIL.

The format of the DO/DU field cards and the method of input is given in sections 3.1.3 and 12.4.4 of the EOD-LARSYS User's document.

If Procedure I is not being exercised, the normal mode of executing the processor remains the same as defined in section 12.0 of the LARSYS document.

3.2.6.4 Outputs

The DISPLAY processor optionally provides an output tape for display on the PMIS DAS (Data Analysis Station).

The DISPLAY processor currently provides the following output to the line printer:

- a line-printer map of each classified field on the input classification results file (MAPTAP).
- classification summaries for each classified field, showing the number of pixels classified into and thresholded from each subclass, class, and category.
- classification summaries of training fields, or optionally test fields. The summaries are presented by subclass, class, and category.

As a result of the modifications described in section 3.2.6, the DISPLAY processor will optionally provide dot data classification performance summaries if the DOTFILE control card described in section 3.2.6.3 is input to the DISPLAY processor.

The additional (and optional) line printer output of dot data performance summaries will be as follows:

- A tabulation of individual dots (pixels) showing the sample/line coordinates of the dot, the labeled category of the dot and the classification category of the dot.
- A classification performance summary by labeled dot data category. This performance summary will include the uncorrected segment (classified field) proportion for each dot data category, the bias corrected segment (classified field) proportion for each dot data category, and the bias correction "alpha" table. In addition, this summary provides the total number of dots (pixels) in each labeled category, the percent correct classification for each category which were thresholded, and the number of dots (pixels) in a given labeled category classified into the labeled category and classified into categories other than the given labeled category.

3.2.6.5 Storage Requirements

TBD

3.2.6.6 Description

The modifications to the DISPLAY processor are such that the analyst may either exercise the present processor capabilities or the LACIE Procedure I capabilities. The difference between the two capabilities is in the type and format of classification performance tables output. See section 3.2.6.4 for a description of the output tables.

The processor will be modified to use existing available storage, normally occupied by training field data, to store the coordinate and category labels, from the input dot data file.

A modification to existing storage allocations will be to increase the maximum number of training and test fields allowable in DISPLAY from the present 200 fields to a maximum of 250 fields.

The subprogram, REDIF3, will be modified to read and de-code the new DOTFILE control card, and to set a flag which indicates that processing of the dot data is to be performed by DISPLAY.

Subprogram DSPLY2 and the utility routines referenced by it, PCT and PRTSUM, will be modified to construct the individual dot data classified category table, using existing storage to construct the table, and also to build the classification summary by labeled dot data category described in section 3.2.6.4.

In order to build this summary, the computations to derive the bias-corrected segment (classified field) proportions, and the related "alpha" table will be performed in DSPLY2. The computations of the uncorrected category proportions, the "alpha" table, and the bias corrected category proportions are as follows:

$$\begin{aligned}
 Pu_i &= \text{uncorrected segment (classified field) proportion} \\
 &\quad \text{for category } C_i \\
 &= \frac{\text{number of dots (pixels) classified as } C_i}{\text{total number pixels in classified field}} \\
 &= \frac{N_{C_i}}{N_T}
 \end{aligned}$$

for $i = 1, 2, \dots$, NOCAT categories of labeled
dot data

$$\text{Alpha}_{i,j} = \alpha_{ij} = \frac{\text{Number of dots labeled } C_j, \text{ classified as } C_i}{\text{Number of dots classified } C_i}$$

for $i, j, = 1, 2, \dots$, NOCAT categories of labeled
dot data

Pc_i = bias corrected proportion for category C_i

$$= \sum_{j=1}^{\text{NOCAT}} (\alpha_{ij} * Pu_j)$$

The computation of the percent correct classification of the
dots will be performed as follows:

$$\frac{\text{Number of dots (pixels) of } C_i \text{ classified as } C_i}{\text{Total number dots in } C_i}$$

$$= \frac{N_{C_i}}{N_{T_{C_i}}}$$

3.2.6.7 Flowchart

N/A

3.2.6.8 Listing

TBD

APPENDIX A

APPENDIX A

TABLE OF CONTENTS

- A.1 Software for Subprogram No. 1 (FLDTYP)
- A.2 Software for Subprogram No. 2 (DOTDAT)
- A.3 Software for Subprogram No. 3 (DOTS)
- A.4 Software for Subprogram No. 4 (DOTFIL)
- A.5 Software for Subprogram No. 5 (SET13)

A.1 SOFTWARE SUBPROGRAM NO. 1 (FLDTYP)

The subprogram FLDTYP initiates the reading of the field data set defined in section 3.2.1.3 and signals the calling routine DOTS when all the fields for a given type have been processed.

A.1.1 LINKAGES

FLDTYP is called by the routine DOTS and calls the routine LAREAD.

A.1.2 INTERFACES

Interfaces with other routines through the common blocks INFORM and DOTVEC.

A.1.3 INPUTS

Calling sequence: CALL FLDTYP(FIELDS, STAMNT, \$, \$, \$, IPT, VERTEX)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
FIELDS	(4,1)	Out	Array for storing the field data:
STAMNT	1	In	Parameter for a go to Fortran statement
\$	1	In	Exit route if a field card is encountered
\$	1	In	Exit route if a TYPE card is encountered

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
\$	1	In	Exit route if a \$END* card is encountered
IPT	1	In	Pointer for FIELDS array
VERTEX	1	Out	Array for storing the field vertices

A.1.4 OUTPUTS

Stores the type and category names in the common block DOTVEC.

A.1.5 STORAGE REQUIREMENTS

TBD

A.1.6 DESCRIPTION

The subprogram FLDTYP signals the calling routine when a TYPE card, field card or \$END* card is encountered by exiting through calling arguments. If a TYPE card is encountered, the subprogram exits by calling argument number 4; if a field card is encountered, the subprogram exits by calling argument number 3; if a \$END* card is encountered, the subprogram exits by calling argument number 5.

A.1.7 FLOW CHART

TBD

A.1.8 LISTING

TBD

A.2 SOFTWARE SUBPROGRAM NO. 2 (DOTDAT)

The subprogram DOTDAT is the driver routine for the DOTDATA processor.

A.2.1 LINKAGES

DOTDAT is called by the monitor routine MONTOR and calls the routines SET13 and DOTS.

A.2.2 INTERFACES

N/A

A.2.3 INPUTS

Calling sequence: CALL DOTDAT (ARRAY, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	10,600	In	Array in blank common dimensioned by the parameter top
TOP	1	In	TOP = 10,600

A.2.4 OUTPUTS

N/A

A.2.5 STORAGE REQUIREMENTS

TBD

A.2.6 DESCRIPTION

The subprogram DOTDAT calls SET13 to read and analyze the control cards. DOTS is called to coordinate the functions required to output the dot data file, DOTFIL.

A.2.7 FLOW CHART

TBD

A.2.8 LISTING

TBD

A.3 SOFTWARE SUBPROGRAM NO. 3 (DOTS)

The subprogram DOTS is the coordinator and supervisor of the functions required to output the dot data file, DOTFIL. This file is an interface for the processors ISOCLS, LABEL and DISPLAY.

A.3.1 LINKAGES

DOTS is called by the routine DOTDAT and calls the routines TAPADR, FLDINT, LINERD, FDLINT, DOTFILE and FLDTYP.

A.3.2 INTERFACES

Stores field and dot vector information in the common block DOTVEC.

A.3.3 INPUTS

Calling sequence: CALL DOTS (DATA, FIELDS, VERTEX, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
DATA	5000	In	Array for storing the dot data.
FIELDS	1000	In	Array for storing the field data
VERTEX	1000	In	Array for storing the field vertices.
TOP	1	In	Parameter. TOP=10,600.

A.3.4 OUTPUTS

Optionally, prints the spatial and spectral information of the dot data file, DOTFIL on the line printer.

A.3.5 STORAGE REQUIREMENTS

TBD

A.3.6 DESCRIPTION

DOTS calls the routine FLDTYP to coordinate the reading of the field cards for a given type. For each field in type i information such as line number, sample number, type number, category number, and dot (grid point) is collected. A maximum of $n \leq 5000 / (\text{number of channels}) \leq 250$ dots may be collected for type i.

When all the fields for type i have been processed, the routine DOTFIL will output the data to a file. A file is written for each type i.

A.3.7 FLOWCHARTS

TBD

A.3.8 LISTING

TBD

A.4 SOFTWARE SUBPROGRAM NO. 4 (DOTFIL)

The subprogram DOTFIL will output the multi-file dot data file, DOTFIL. The processors ISOCLS and LABEL will read the file containing type 1 dots; the processor DISPLAY will read the file containing type 2 dots.

The dots in the file may be labeled or relabeled, but not retyped. To retype a dot, the file must be constructed again by executing the DOTDATA processor.

A.4.1 LINKAGES

DOTFIL is called by the routine DOTS and does not call any LARSYS routines.

A.4.2 INTERFACES

Interfaces with other routines through the common blocks GLOBAL, INFORM, and DOTVEC.

A.4.3 INPUTS

Call sequence: CALL DOTFILE (DATA, FIELDS, VERTEX)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
DATA	(4+NOFEAT,1)	In	Array containing the dot data.
FIELDS	(4,1)	In	Array containing the field data.
VERTEX	(2,1)	In	Array containing the field vertices.

A.4.4 OUTPUTS

A multi-file, unformatted FORTRAN written file is output. The default unit for outputting the file is logical unit F (FORTRAN unit 8). See Appendix G for format of the file.

A line printer summary of the input field information will be printed.

A.4.5 STORAGE REQUIREMENTS

TBD

A.4.6 DESCRIPTION

For each call to DOTFIL a file will be output and a summary of the field information will be printed.

An unformatted FORTRAN write statement will create the records with three records comprising a file. The files will be written sequentially.

A.4.7 FLOWCHARTS

TBD

A.4.8 LISTING

TBD

A.5 SOFTWARE SUBPROGRAM NO. 5 (SET13)

The subprogram SET13 reads and analyzes the control cards for the DOTDATA processor.

A.5.1 LINKAGES

SET13 is called by DOTDAT and calls the routines NUMBER, ORDER, NXTCHR, and FIND.

A.5.2 INTERFACES

Interfaces with other routine through the common blocks GLOBAL, INFORM, and DOTVEC.

A.5.3 INPUTS

- Calling sequence: CALL SUBROUTINE SET13
(No calling arguments are used)
- Control Cards
For description of control cards acceptable by this routine, see section 3.2.1.3.

A.5.4 OUTPUTS

A line printer summary of the selected options is printed.

Parameter values such as file switches and option switches will be stored in the common blocks.

A.5.5 STORAGE REQUIREMENTS

TBD

A.5.6 DESCRIPTION

Each control card will be read and analyzed for format errors. If the control card is valid, the appropriate parameters and

switches will be set. When applicable, if a control card for a specific option is not input, a default value will be supplied.

A.5.7 FLOWCHARTS

TBD

A.5.8 LISTING

TBD

APPENDIX B

APPENDIX B

TABLE OF CONTENTS

B.1	Software for Subprogram No. 1 (ALLKIN)
B.2	Software for Subprogram No. 2 (CNDMAP)
B.3	Software for Subprogram No. 3 (DOTDST)
B.4	Software for Subprogram No. 4 (KNEAR)
B.5	Software for Subprogram No. 5 (LABDOT)
B.6	Software for Subprogram No. 6 (LABEL)
B.7	Software for Subprogram No. 7 (LABLR)
B.8	Software for Subprogram No. 8 (LABMAN)
B.9	Software for Subprogram No. 9 (MIXMAP)
B.10	Software for Subprogram No. 10 (SET14)

B.1 SOFTWARE SUBPROGRAM NO. 1 (ALLKIN)

The subprogram ALLKIN labels the statistics obtained from the clustering processor ISOCLS by all-of-a-kind procedure.

B.1.1 LINKAGES

The subprogram ALLKIN is called by the routine LABLR and if applicable, calls the routine KNEAR.

B.1.2 INTERFACES

Interfaces with other routines through the common blocks INFORM and LABS.

B.1.3 INPUTS

Calling sequence: CALL ALLKIN (DIST, MEANS, DOTARY, LABMEAN,
DOTNOS)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Function</u>
DIST	(5,250)	In	Array containing the distances.
MEANS	(NOFEAT,NOSUB2)	In	Array containing the unlabeled means.
DOTARY	(4+NOFEAT,250)	In	Array containing the dot data.
LBMEAN	(NOFEAT,NOSUB2)	Out	Output array containing the labeled means.
DOTNOS	(K,NOSUB2)	In	Dot number for the corresponding distances in DIST.

B.1.4 OUTPUTS

A line printer summary of the labeling dots within a cluster will be printed.

B.1.5 STORAGE REQUIREMENTS

TBD

B.1.6 DESCRIPTION

For a given cluster, the array DOTARY will be scanned and all the dots assigned to that cluster during the clustering process will be grouped.

The labels for the group of dots will be scanned. If the labels are of one category, the cluster will be labeled that category. If more than one category is encountered, the category with the majority of the dots will label the cluster, and the cluster will be flagged as a conditional cluster.

If a given cluster does not contain any labeling dots within the cluster, the k-nearest neighbor procedure will be applied.

B.1.7 FLOWCHARTS

TBD

B.1.8 LISTING

TBD

B.2 SOFTWARE SUBPROGRAM NO. 2 (CNDMAP)

The subprogram CNDMAP outputs a conditional cluster map.

B.2.1 LINKAGES

The subprogram CNDMAP is called by the routine LABLR and calls the routines WRTHEd and WRTLN.

B.2.2 INTERFACES

Interfaces with other routines through the common blocks INFORM, GLOBAL and LABS.

B.2.3 INPUTS

Calling sequence: CALL CNDMAP (MEANS, CNDCLS, NOCLST)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
MEANS	(NOFEAT,NOSUB2)	In	Array containing the unlabeled means.
CNDCLS	60	In	Array containing the cluster number of the conditional cluster.
NOCLST	1	In	Number of conditional clusters.

B.2.4 OUTPUTS

A conditional cluster map will be output on the analyst specified unit. The default unit will be FORTRAN unit 16 (logical unit N). (See section 5.1 in LARSYS document for format of file).

A line printer symbol map of the conditional cluster map will be printed.

B.2.5 STORAGE REQUIREMENTS

TBD

B.2.6 DESCRIPTION

The MAPFIL will be read into core from the drum a line at a line. The line will be scanned for the cluster number of any cluster(s) that is to be flagged as conditional. If such a cluster number is found, the existing cluster number will be changed to (NOSUB2+1).

In creating the spatial conditional map, when a cluster number greater than the number of existing clusters is encountered, the color code representing a conditional cluster will be output.

B.2.7 FLOWCHART

TBD

B.2.8 LISTING

TBD

B.3 SOFTWARE SUBPROGRAM NO. 3 (DOTDST)

The subprogram DOTDST computes the inter-cluster-dot distance. For each cluster, only the k-nearest labeling dots are saved in the table.

B.3.1 LINKAGES

The subprogram DOTDST is called by the routine LABLR and calls the routine SORTVC.

B.3.2 INTERFACES

Interfaces with other routines through the common blocks INFORM and LABS.

B.3.3 INPUTS

Calling sequence: CALL DOTDST (DOTARY, MEANS, DIST)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
DOTARY	(NOFEAT+4,250)	In	Array containing the dot data.
MEANS	(NOFEAT,NOSUB2)	In	Array containing the unlabeled means.
DIST	(K,NOSUB2)	Out	Output array containing the distances of the k-nearest dots for each cluster.

B.3.4 OUTPUTS

The inter-distance table will be printed on the line printer.

B.3.5 STORAGE REQUIREMENTS

TBD

B.3.6 DESCRIPTION

For the dots in DOTARY, the inter-distance between each dot and a given cluster is computed using either the L_1 or L_2 distance. The distances are stored in a linear array. This array is sorted in descending order and only the first k distances and corresponding dot numbers are saved in the distance table.

B.3.7 FLOWCHART

TBD

B.3.8 LISTING

TBD

B.4 SOFTWARE SUBPROGRAM NO. 4 (KNEAR)

The subprogram KNEAR will label the statistics obtained from the clustering processor ISOCLS by the k-nearest neighbor procedure.

B.4.1 LINKAGES

The subprogram KNEAR is called by the routine LABLR and does not call any LARSYS routines.

B.4.2 INTERFACES

Interfaces with other routines through the common blocks INFORM and LABS.

B.4.3 INPUTS

Calling sequence: CALL KNEAR (DIST, MEANS, DOTARY, LABMEAN, DOTNOS)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
DIST	(5,250)	In	Array containing the distances.
MEANS	(NOFEAT,NOSUB2)	In	Array containing the unlabeled means.
DOTARY	(NOFEAT+4,250)	In	Array containing the dot data.
LBMEAN	(NOFEAT,NOSUB2)	Out	Output array containing the labeled means.
DOTNOS	(K,NOSUB2)	In	Array containing the dot number corresponding to the distances in DIST.

B.4.4 OUTPUTS

A line printer summary of the k-nearest labeling dots will be printed.

B.4.5 STORAGE REQUIREMENTS

TBD

B.4.6 DESCRIPTION

For a given cluster, a poll of the category labels of the k-nearest dots to the cluster is computed. The label with the majority of the dots will label the cluster. If the distance between the cluster mean and the nearest identically labeled dot is greater than the threshold value t , the cluster is flagged as conditional.

B.4.7 FLOWCHART

TBD

B.4.8 LISTING

TBD

B.5 SOFTWARE SUBPROGRAM NO. 5 (LABDOT)

The subprogram LABDOT labels or relabels the dot data file DOTFIL and outputs the updated file to tape.

B.5.1 LINKAGES

The subprogram LABDOT is called by the routine LABLR and calls the routine DOTWRT.

B.5.2 INTERFACES

Interfaces with other routines through the common blocks INFORM, GLOBAL and LABS.

B.5.3 INPUTS

Calling sequence: CALL LABDOT (NOCAT2, CATNM2, CATPRT, CATDOT)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
NOCAT2	1	In	Number of category names being input.
CATNM2	60	In	Array containing the category names for updating DOTFIL.
CATPTR	120	In	Array containing the pointers into the array CATDOT.
CATDOT	60	In	Array containing the dot number for the dots to be re-labeled.

B.5.4 OUTPUTS

An undated file containing the relabeled dots is output on an analyst specified unit. The default unit is FORTRAN unit 8 (logical unit F). (See Appendix G for format of file)

B.5.5 STORAGE REQUIREMENTS

TBD

B.5.6 DESCRIPTION

The category names input through control cards are checked for new name entries. If a new category has been input, the arrays concerning the category information is adjusted.

Each dot number input through a control card in the setup routine is relabeled to the appropriate label. No dots may be deleted from the set of dots being updated.

After the relabeling process is finished, the updated file is written.

B.5.7 FLOWCHARTS

TBD

B.5.8 LISTING

TBD

B.6 SOFTWARE SUBPROGRAM NO. 6 (LABEL)

The subprogram LABEL is the driver routine for the LABEL processor.

B.6.1 LINKAGES

The subprogram LABEL is called by the monitoring routine MONTOR and calls routines SET14 and LABLR.

B.6.2 INTERFACES

Interfaces with other routine through the common blocks INFORM and LABS.

B.6.3 INPUTS

Calling sequence: CALL LABEL (ARRAY, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	10,600	In	Array in blank common.
TOP	1	In	Parameter. TOP=10,600

B.6.4 OUTPUTS

N/A

B.6.5 STORAGE REQUIREMENTS

TBD

B.6.6 DESCRIPTION

The setup routine, SET14 is called to read and analyze the control cards. The routine LABLR will coordinate the functions required to execute the options specified by the analyst.

The routine DISMAP will output the cluster map in the format acceptable to the DISPLAY processor.

B.6.7 FLOW CHART

TBD

B.6.8 LISTING

TBD

B.7 SOFTWARE SUBPROGRAM NO. 7 (LABLR)

The subprogram LABLR is the coordinator and supervisor of the routines required to perform the operations specified by the analyst.

The analyst may either update the DOTFIL and/or SAVTAP files or label a given set of statistics by one of two procedures (k-nearest neighbor or all-of-a-kind) in one execution of the LABEL processor.

B.7.1 LINKAGES

The subprogram LABLR is called by the routine LABEL and calls routines LABMAN, LABDOT, KNEAR, ALLKIN, CNDMAP, and MIXMAP.

B.7.2 INTERFACES

Interfaces with routines through the common blocks INFORM, GLOBAL, and LABS.

B.7.3 INPUTS

Calling sequence: CALL LABLR (ARRAY, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	10,600	In	Array in blank common
TOP	1	In	Parameter. TOP=10,600.

B.7.4 OUTPUTS

N/A

B.7.5 STORAGE REQUIREMENTS

TBD

B.7.6 DESCRIPTION

Depending on the analyst selected options LABLR supervises the following functions:

- Reading of the MAPFIL (unconditional cluster map)
- Relabeling of the SAVTAP file
- Relabeling of the DOTFIL file
- Execution of the k-nearest neighbor procedure
- Execution of the all-of-a-kind procedure
- Output of the conditional cluster map
- Output of the mixed cluster map

B.7.7 FLOW CHART

TBD

B.7.8 LISTING

TBD

B.8 SOFTWARE SUBPROGRAM NO. 8 (LABMAN)

The subprogram LABMAN relabels the statistics file SAVTAP by input of the control card STATLABEL and outputs the updated file to tape.

B.8.1 LINKAGES

The subprogram LABMAN is called by the routine LABLR and calls the routine REORDER.

B.8.2 INTERFACES

Interfaces with other routines through the common blocks INFORM and LABS.

B.8.3 INPUTS

Calling sequence: CALL LABMAN(CLSNM2, NOCLS2, SUBRAY, PTR)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
CLSNM2	60	In	Array containing the classnames for updating the SAVTAP file.
NOCLS2	1	In	Number of class names being input.
SUBARY	60	In	Array containing the cluster numbers of the clusters to be relabeled.
PTR	120	In	Pointer for the array SUBRAY.

B.8.4 OUTPUTS

An updated file containing the manually relabeled statistics is output on an analyst specified unit. The default unit is FORTRAN unit 1 (logical unit A), file 1. (See section 4.1 of LARSYS document for format of file).

B.8.5 STORAGE REQUIREMENTS

TBD

B.8.6 DESCRIPTION

The analyst may manually relabel the statistics file by input of control cards. The classnames (categories) input must match a name on the statistics file.

The analyst may regroup any clusters into another class. For example, if the cluster WHT1 belonged to the class WHEAT and the analyst decided that the cluster should belong to the NONWHT class, he may force the cluster WHT1 to be assigned to the class NONWHT.

As a result of regrouping the clusters, the order of the statistics on the file will be rearranged, but the regrouped cluster will not be renamed.

B.8.7 FLOWCHART

TBD

B.8.8 LISTING

TBD

B.9 SOFTWARE SUBPROGRAM NO. 2 (MIXMAP)

The subprogram MIXMAP outputs a conditional cluster map.

B.9.1 LINKAGES

The subprogram MIXMAP is called by the routine LABLR and calls the routines WRTHEd and WRTLN.

B.9.2 INTERFACES

Interfaces with other routines through the common blocks INFORM, GLOBAL and LABS.

B.9.3 INPUTS

Calling sequence: CALL MIXMAP (MEANS, NXCLS, NCLSTR)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
MEANS	(NOFEAT,NOSUB2)	In	Array containing the unlabeled means.
MIXCLS	60	In	Array containing the cluster number of the mixed cluster.
NCLSTR	1	In	Number of mixed clusters.

B.9.4 OUTPUTS

A mixed cluster map will be output on the analyst specified unit. The default unit will be FORTRAN unit 16 (logical unit N). (See section 5.1 in LARSYS document for format of file).

A line printer symbol map of the mixed cluster map will be printed.

B.9.5 STORAGE REQUIREMENTS

TBD

B.9.6 DESCRIPTION

The MAPFIL will be read into core from the drum a line at a line. The line will be scanned for the cluster number of any cluster(s) that is to be flagged as mixed. If such a cluster number is found, the existing cluster number will be changed to (NOSUB2+1).

In creating the spatial mixed map; when a cluster number greater than the number of existing clusters is encountered, the color code representing a mixed cluster will be output.

B.9.7 FLOWCHART

TBD

B.9.8 LISTING

TBD

B.10 SOFTWARE SUBPROGRAM NO. 10 (SET14)

The subprogram SET14 reads and analyzes the control cards for the LABEL processor.

B.10.1 LINKAGES

The subprogram SET14 is called by the driver routine LABEL and calls the routines NUMBER, ORDER, NXTCHR, FIND, and RDDOTS.

B.10.2 INTERFACES

Interfaces with other routines through the common blocks GLOBAL, INFORM, and LABS.

B.10.3 INPUTS

- Calling sequence: CALL SET14 (ARRAY, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	10,600	In	Array in blank common.
TOP	1	In	Parameter.TOP=10,600

- Control Cards

For description of control cards acceptable by this routine, see section 3.2.2.3.

B.10.4 OUTPUTS

A line printer summary of the selected options is printed.

Parameter values such as file switches and option switches will be stored in the common blocks INFORM, GLOBAL, and LABS.

B.10.5 STORAGE REQUIREMENTS

TBD

B.10.6 DESCRIPTION

Each control card will be read and analyzed for format errors. If the control card is valid, the appropriate parameters and switches will be set. When applicable, if a control card for a specific option is not input, a default value will be supplied.

If applicable, the dot data file, DOTFIL, will be read in and all the dots input on the EXCLUDE control card will be flagged as belonging to category 0. In executing one of the procedures, all dots of category 0 will be excluded from all computations.

B.10.7 FLOWCHART

TBD

B.10.8 LISTING

TBD

APPENDIX C

APPENDIX C

TABLE OF CONTENTS

- C.1 Software for Subprogram No. 1 (ISODAT)
- C.2 Software for Subprogram No. 2 (RDDATA)
- C.3 Software for Subprogram No. 3 (SETUP7)
- C.4 Software for Subprogram No. 4 (TAPERD)

C.1 SOFTWARE SUBPROGRAM NO. 1 (ISODAT)

The new capabilities of subprogram ISODAT are:

1. on option, call the utility routine RDDOTS to obtain cluster starting vectors,
2. on option, apply a sun angle correction by channel to each pixel prior to clustering,
3. remove "designated other" or "designated unidentifiable" pixels from the clustering algorithm input.

C.1.1 LINKAGES

The Subprogram ISODAT is called by the routine ISOCLS, and calls routines PRINT, PSPLIT, CLDIST, and DELETE.

C.1.2 INTERFACES

Subprogram ISODAT interfaces with other routines through common blocks GLOBAL, PASS, and ISOLNK.

C.1.3 INPUTS

- Calling sequence: CALL ISODAT (C, IPLACE, MEANS, N, STDEV, CLD, FLDINF, AVP, AMN)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
C	(NOFEAT,NOPTS)	Out	Starting location for one drum read of pixels for clustering
IPLACE	NOPTS	Out	NOPTS is the number of points for one read starting location for one drum write of pixel cluster numbers.

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
MEANS	(NOFEAT, MAXCLS)	In/Out	Starting mean values for clustering, MAXCLS is the maximum number of clusters.
N	MAXCLS	Out	Number of pixels per cluster.
STDEV	(NOFEAT, MAXCLS)	Out	Standard deviations per cluster/channel.
CLP	(MAXCLS, MAXCLS)	In	Cluster distance matrix.
FLDINF	(IPT)	Out	Field information (vertices, vertex location).
AVP	(NOFEAT, MAXCLS)	Out	Temporary storage for computation of cluster means
AMN	(NOFEAT, MAXCLS)	In	Temporary storage for computation of cluster standard deviations.
DODUF	11500 -2* MAXCLS*NOFEAT -MAXCLS*MAXCLS	In	DO/DU pixel locations on drum file.

C.1.4 OUTPUTS

No additional outputs.

C.1.5 STORAGE REQUIREMENTS

TBD

C.1.6 DESCRIPTION

A dot file trigger in labeled common ISOLNK will invoke a call to RDDOTS for starting vectors. The unit and file numbers will have been loaded in labeled common GLOBAL. A sun angle correction trigger in labeled common ISOLNK will cause the sun angles (stored in ISOLNK) to be used to find sun angle corrections from an internal table. These corrections will be stored in ISOLNK indexed by channel number. Application of the correction consists of multiplying the pixel channel radiance value by the correction.

Using the drum addresses stored at DODUF and computed in RDDATA, DO/DU pixels will be deleted from input to the clustering computations. Prior to the cluster numbers being written to drum, the special DO or DU cluster number will be assigned to DO or DU pixels.

C.1.7 FLOW CHART

N/A

C.1.8 LISTING

TBD

C.2 SOFTWARE SUBPROGRAM NO. 2 (RDDATA)

The new capabilities of subprogram RDDATA are:

1. read DESIGNATED OTHER and DESIGNATED UNIDENTIFIABLE class-name cards and the associated field definition cards, and calculate the field information (number of vertices, vertex locations, number of pixels) for both classes,
2. compute information concerning the DO/DU pixels for use by other subprograms. This includes: the number of DO/DU pixels in each pixel field for the current class, and drum addresses for DO/DU pixels,
3. assign special cluster numbers and printing symbols to DO and DU pixels, as well as special mean values (0 or 255).

C.2.1 LINKAGES

The subprogram RDDATA is called by the routine ISOCLS, and calls the routines TAPERD and LAREAD.

C.2.2 INTERFACES

Subprogram RDDATA interfaces with other routines through common blocks GLOBAL, PASS, and ISOLNK.

C.2.3 INPUTS

- Calling Sequence: CALL RDDATA (ARRAY, TOP, IDATA, IDIM, LAST)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	TOP	In	Starting location for field information.
TOP	1	In	Number of available words in ARRAY.

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
IDATA	IDIM	In	Buffer for pixel value storage per scan line.
IDIM	1	In	Maximum dimension of IDATA.
LAST	1	Out	Flag indicating (when set to a value of 1) that all classes for one set of control cards have been processed.

C.2.4 OUTPUTS

No additional outputs.

C.2.5 STORAGE REQUIREMENTS

TBD

C.2.6 DESCRIPTION

Field information for designated "other" and "unidentifiable" pixel fields will be processed and made available to ISODAT through the calling sequence variable KVAR. This will be done after all drum write buffering is completed.

The information will consist of:

1. number of DO/DU pixels per field per class
2. drum location of DO/DU pixels.

~~C-6~~
91

The field description from cards will be held internally.

The trigger for DO/DU processing will be added to labeled common PASS and set in RDDATA. Two special print symbols for DO and DU pixels will be added to the symbol vector SYMMTX in labeled common PASS. (This will require re-dimensioning SYMMTX).

Special "cluster" numbers will be assigned to DO and DU pixels. These two numbers will appear in ISOLNK labeled common. Special mean values will also be set.

C.3 SOFTWARE COMPONENT NO. 3 (SETUP7)

The new capabilities of subprogram SETUP7 are:

1. read an optional DOTFILE control card with unit and file number, and set option trigger for subsequent use by ISODAT
2. read an optional SUNANG control card, set the sun angle correction trigger for use by ISODAT, and extract the sun angles if these are present on the card.

C.3.1 LINKAGES

The subprogram SETUP7 is called by the routine ISOCLS. Routine SETUP7 calls card reader/interpreter routines FIND, NXTCHR, NUMBER, and FLTNUM.

C.3.2 INTERFACES

Subprogram SETUP7 interfaces with other routines through common blocks GLOBAL, PASS, and ISOLNK.

C.4.3 INPUTS

- Calling sequence: CALL SETUP7 (ARRAY, TOP, ITIME)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	TOP-CORBAS	Out	Stores means or a statistics Module STAT deck.
TOP	1	In	Maximum dimension of ARRAY (normally 10,600)
ITIME	1	In	Count of number of times SETUP7 called

● New Control Cards:

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
DOTFIL	UNIT=n, FILE=M (Default: Self- initialized start- ing)	Defines the FORTRAN unit number n and file number m of the dot data file DOTFIL con- taining the starting vectors.
SUNANG	TAPE (Default: No sun angle correction applied)	Sun angles will be ex- tracted from the MSS data tape.
SUNANG	n_1, n_2, \dots, n_j n_j are integers, $j \leq 7$ (Default: No sun angle correction applied)	$\{n_j\}$ are the sun angles to be used in computing the sun angle correc- tions for use in the clustering algorithm. A sun angle must be in- put for each set of 4 channels appearing on the CHANNEL control card.

C.3.4 OUTPUTS

No additional outputs.

C.3.5 STORAGE REQUIREMENTS

TBD

C.3.6. DESCRIPTION

The DOTFILE and SUNANG control cards will be handled by ex-
tending data vector INVEC to include the character strings

DOTFIL and SUNANG. The unit and file number of DOTFILE will be stored in GLOBAL labeled common. Sun angles appearing on the SUNANG card will be placed in ISOLNK as sun angle card input. The sun angle correction trigger will appear in ISOLNK and will be set by SETUP7.

C.3.7 FLOW CHART

N/A

C.3.8 LISTING

TBD

C.4 SOFTWARE SUBPROGRAM NO. 4 (TAPERD)

The new capabilities of subprogram TAPERD are:

1. on option, unpack sun angles from the header record of the MSS Universal format input tape,
2. unpack the stop pixel number, pixel skip number, and line skip number from the header record of the MSS Universal format input tape.

C.4.1 LINKAGES

The subprogram TAPERD is called by the routine RDDATA.

C.4.2 INTERFACES

The subprogram TAPERD interfaces with other routines through common block ISOLNK.

C.4.3 INPUTS

TAPERD has 3 entry points: TAPHRR, FLDINT and LINERD

- Calling sequence: CALL TAPHDR (DATAPE, DATFIL)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
DATAPE	1	In	Tape unit number.
DATFIL	1	In	Number of EOF's to be read over in positioning tape.

- Calling sequence: CALL FLDINT (FLDINF, FETVEC, NOFEAT)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
FLDINF	6	In	Rectangular field description for use in unpacking pixels from MSS tape.

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
FETVEC	30	In	Contains channels requested.
NOFEAT	1	In	Number of channels in FETVEC.

● Calling sequence: CALL LINERD (IDATA, ENDTAP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
IDATA	KVARDM (=11500)	Out	Array containing unpacked data.
ENDTAP	1	Out	EOF trigger: returns value of -1 if an EOF found.

C.4.4 OUTPUTS

N/A

C.4.5 STORAGE REQUIREMENTS

TPD

C.4.6 DESCRIPTION

The information to be unpacked from the Universal format MSS tape occupies the following byte positions on the tape header record:

	<u>Byte Numbers</u>	<u>Description</u>
	110- 111	pixel stop number
	1789-1790	pixel skip factor
	1791-1792	line skip factor
Tape Sun Angles {	2201-2202	sun angle for pass 1 (channels 1-4)
	2203-2304	sun angle for pass 2 (channels 5-8)
	etc.	etc. (up to 30 channels)

This information will be unpacked by simply extending the TAPHDR routine data vectors HWRD, BIT and NB, expanding the dimension of the ID vector, and expanding an existing unpacking code block. The information listed above, plus the start pixel number, will be placed in labeled common ISOLNK.

C.4.7 FLOW CHART

N/A

C.4.8 LISTING

TBD

APPENDIX D

APPENDIX D

TABLE OF CONTENTS

- D.1 SOFTWARE FOR SUBPROGRAM NO. 1 (SETUP4)
- D.2 SOFTWARE FOR SUBPROGRAM NO. 2 (WGTCHK)

D.1 SOFTWARE COMPONENT NO. 1 (SETUP4)

The SETUP4 subprogram of the SELECT processor will recognize the new option, "CLSWT", on the OPTION control card. If the new option is input, SETUP4 will initialize the table of weights for interclass subclass pairs (=1.0) and for intra-class subclass pairs (=0.0).

D.1.1 LINKAGES

The SETUP4 subprogram is called from the processor driver program, SELECT. Subprogram SETUP4 calls the following subprograms: BMFIL, BSTCHK, CRDSTA, EVLCHK, FIND, GRPSCN, NUMBER, NXTCHR, ORDER, PRTFLD, REDSAV, WGTCHK, and WGTSCN.

D.1.2 INTERFACES

SETUP4 interfaces with the driver program, SELECT, and other programs via the named common blocks FSL, GLOBAL, and INFORM.

D.1.3 INPUTS

- Calling sequence: CALL SETUP4 (ARRAY, TOP, STOPFG, JTIME, SUBRAY, SUBSIZ)

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
ARRAY	10600	In	Working array in blank COMMON.
TOP	1	In	The maximum usable location in ARRAY. TOP=10600, set in MONITOR.
STOPFG	1	Out	Flag set=1 to indicate that the \$END* card has been read.

<u>Parameter</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
JTIME	1	In	Counter for the number of times SETUP4 is called from SELECT. When JTIME=1, the line printer output is labeled with the HEADER array contents. For JTIME≠1, the heading is not placed on the output.
SUBRAY	12000	In	Working array, provided from the SELECT program.
SUBSIZ	1	In	Maximum usable location in SUBRAY, set in SELECT program (=12000).

- Revised Control Card

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
(col. 1)	(col. 11-72)	
OPTION	CLSWT	SETUP4 sets a flag, WTKEY=1, to indicate that SETUP4 is to initialize the table of weights for interclass subclass pairs.

D.1.4 OUTPUTS

The modifications to SETUP4 will not affect the current line printer or file output descriptions.

D.1.5 STORAGE REQUIREMENTS

TBD

D.1.6 DESCRIPTION

To establish recognition of the new option, "CLSWT", SETUP4 modifications will involve re-dimensioning the CODVEC array, adding "CLSWT" to the CODVEC array, and providing a transfer point at which a flag, WTKEY, will be set (WTKEY=1) when the character "C" is encountered on the OPTION control card.

After all control cards have been read by SETUP4, the flag WTKEY will be used to either initiate (WTKEY=1) the table or bypass (WTKEY=0) the table of subclass pair weights, preserving the interclass associations of the subclass pairs initialized.

Coding which implements the computation of weight table addresses for interclass subclass pairs will be added to the SETUP4 subprogram. Storage for the weight table will utilize existing storage currently used as a weight table, beginning at ARRAY (WGHS14). The weight table will be initialized =1.0 for all interclass subclass pairs, =0.0 for all intraclass subclass pairs.

The flag, WTKEY, will be added to the calling argument list in the call to the WGTCHK subprogram.

D.1.7 FLOWCHART

N/A

D.1.8 LISTING

TBD

D.2 SOFTWARE COMPONENT NO. 2 (WGTCHK)

Subprogram WGTCHK of the SELECT processor, when called due to one or more "WEIGHTS" cards being read by SETUP4, will perform one of two functions:

- a. The currently available capability of initializing all intersubclass pair weights =1.0, followed by overriding the preset weights with the input weights for specific subclass pairs (and honoring the "OTHERS" weight assignments, if input), or
- b. Directly storing into the weight table only the input weight(s) for specific subclass pair(s), bypassing the initialization of the weight table as is currently done (and ignoring the "OTHERS" weight assignment, if input).

D.2.1 LINKAGES

Subprogram WGTCHK is called by the SETUP4 subprogram. WGTCHK does not reference any lower level subprograms.

D.2.2 INTERFACES

Subprogram WGTCHK does not use a common block. Interface with the SETUP4 subprogram is via the calling arguments.

D.2.3 INPUTS

- Calling sequence: CALL WGTCHK(WEIGHT,CLSNAM,NAMPR, WGHT,WPTR,WRKRY,NOCLS2,WTKEY)

<u>Parameters</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
WEIGHT	NOCLS2* (NOCLS2-1)/2 (NOCLS2=No. of subclasses)	In/Out	Table of weights for subclass pairs. WEIGHT(1) is equivalent to ARRAY(WGHS14) in the calling program SETUP4.

<u>Parameters</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
CLSNAM	NOCLS2	In	Final set of subclass names from SAVTAP file, after any grouping of subclasses performed due to GROUP control card.
NAMPR	(2,WPTR)	In	Subclass names or name pairs input to SETUP4 on "WEIGHTS" card(s). NAMPR(1,WPTR)=subname ₁ , NAMPR(2,WPTR)=subname ₂ , if input, otherwise=blank. NAMPR(1,WPTR) may also contain "OTHERS", from "WEIGHTS OTHERS" card.
WGHT	400	In	Subclass, subclass pair, or "OTHERS" weights from "WEIGHTS" card(s) input to SETUP4 via the WGTSCN subprogram.
WPTR	1	In	The number of weights input on WEIGHTS control card(s).
WRKRY	(NOCLS2, NOCLS2) NOCLS2=No. of subclasses	Out	Working array containing initialization weights for intersubclass pairs (=1.0), and interim storage of input subclass pair weights, from WGHT array.
NOCLS2	1	In	The total number of subclasses for all classes.

<u>Parameters</u>	<u>Dimension</u>	<u>In/Out</u>	<u>Definition</u>
WTKEY	1	In	Flag from SETUP4. If WTKEY=1, indicates that subclass pair weights have been initialized in SETUP4. WGTCHK stores into WEIGHT table only the weights for specific subclass pairs in NAMPR array. If WTKEY=0, WGTCHK performs the usual initialization of weights for all subclass pairs to be =1.0, (or ="OTHERS" weight), then proceeds to override the initialization for subclass pairs in the NAMPR array, with weights from the WGHT array.

D.2.4 OUTPUTS

The modifications to WGTCHK make no change in currently available line printer or file output.

D.2.5 STORAGE REQUIREMENTS

TBD

D.2.6 DESCRIPTION

The calling sequence of WGTCHK will be modified to add the WTKEY flag from SETUP4. A test on the value of the flag will be performed.

If WTKEY=0, subprogram WGTCHK will perform its current function; i.e., pre-initializing all intersubclass weights either =1.0, or set to the value on the "WEIGHTS OTHERS" card, if it is input to SETUP4. Following pre-initialization in the working array, WRKRY, the subprogram currently overrides the preset weights in the working array with weights from the WGHT array using subclass names in the NAMPR array to direct the storage of input weights. When the working array is completed, the values from the working array are transferred to the weight table (WEIGHT) and output by WGTCHK via calling argument.

If WTKEY=1, subprogram WGTCHK will be modified to pre-initialize the weights =0.0 for all subclass pairs in the working array, WRKRY, and to ignore "OTHERS" if it appears in the NAMPR array.

Any other subclass name pairs in the NAMPR array will cause the corresponding weight from the WGHT array to be stored in the working array, for the given subclass pair. When all (WPTR) subclass name pairs in NAMPR have been processed, subclass pair weights will be transferred from the working array to the correct locations in the weight table (WEIGHT), only for those subclass pairs in the working array having a weight greater than 0.0. This will preserve the initialization of interclass subclass pair weights performed in SETUP4, except for specific subclass pair weights set by the user via the "WEIGHTS" card.

D.2.7 FLOWCHART

N/A

D.2.8 LISTING

TBD

APPENDIX E

APPENDIX E

TABLE OF CONTENTS

- E.1 SOFTWARE FOR SUBPROGRAM NO. 1 (REDIF2)
- E.2 SOFTWARE FOR SUBPROGRAM NO. 2 (SETUP2)

E.1 SOFTWARE COMPONENT NO. 1 (REDIF2)

Subprogram REDIF2 of the CLASSIFY processor will be modified to, (1) accept a revised APRIORI control card and (2) set a flag (APRFLG) to indicate that in the subprogram SETUP2, the apriori probability values for each subclass are to be computed, using subclass population data from the statistics file, SAVTAP.

E.1.1 LINKAGES

Subprogram REDIF2 is called by subprogram SETUP2. Subprogram REDIF2 calls the following subprograms:

BMFIL, CATSCN, CRDSTA, FIND, FLTNUM, GRPSCN, NUMBER, NXTCHR, ORDER, and TDATE.

E.1.2 INTERFACES

The common blocks used by subprogram REDIF2 are: CLASS, GLOBAL, INFORM. The variable, APRFLG, in common block CLASS will be involved in the modification to subprogram REDIF2.

E.1.3 INPUTS

- Calling sequence: Call REDIF2 (ARRAY, TOP, APRIOR, KATNO, BMATRX, PRIORI)

<u>Parameters</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
ARRAY	10 600	In	Variably dimensioned working storage
TOP	1	In	Maximum usable location in ARRAY - TOP = 10, 600
APRIOR	60	In	Storage for a priori probability values - Initialize = 0.0 in REDIF2

<u>Parameters</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
KATNO	60	In	Unused in REDIF2.
BMATRX	(BMCOMB, BMFEAT) BMCOMB = no. of linear combinations in B-matrix BMFEAT = no. of channels used in B-matrix	Out	Storage location for B-matrix
PRIORI	60	Out	Storage for apriori probability values read from input APRIORI control card.

● Revised control card

<u>Keyword</u>	<u>Parameter</u>	<u>Function</u>
APRIORI	FILE	If the first nonblank character on the "APRIORI" card (parameter field) is an "F", the flag, APRFLG, is set = -777777. This value of APRFLG will initiate in SETUP2 the computation of subclass apriori probability values from the statistics file's populations data.

E.1.4 OUTPUTS

File output will be unchanged by these modifications to REDIF2.

Line printer output will be modified to include the message
"*** CLSFY/REDIF2 - BAD INPUT ON APRIORI CARD - DEFAULT
APRIORI PROBABILITY VALUES WILL BE USED." This message
will be printed immediately following the listing of the
APRIORI control card as is currently done, and is only
printed when the first character on the APRIORI control card
is neither a legitimate "F", nor a number in the range of
0, 1, ..., 9.

E.1.5 STORAGE REQUIREMENTS

TBD

E.1.6 DESCRIPTION

REDIF2 modifications will include: (1) a test for the character
"F" on the APRIORI control card, (2) the initialization of the
variable, APRFLG = -777777 if the character "F" is encountered.

If the first nonblank character is not an "F", REDIF2 will
proceed to the currently available reading of apriori proba-
bility values or if the first character is not a legitimate
numeric number, an error message will be printed (see sec-
tion E.1.4). In the case of an error on the APRIORI control
card the variable, APRKEY is set = 0.

E.1.7 FLOW CHART

N/A

E.1.8 LISTING

TBD

E.2 SOFTWARE COMPONENT NO. 2 (SETUP2)

When modified, subprogram SETUP2 will test the value of the flag, APRFLG, which is returned from subprogram REDIF2 and if APRFLG = -777777, SETUP2 will use subclass population data obtained from the statistics file, SAVTAP, to compute the subclass apriori probability values which will be stored in the APRIOR array.

E.2.1 LINKAGES

Subprogram SETUP2 is called by the CLASSIFY processor driver program, CLSFY. Subprogram SETUP2 calls the following subprograms: FIND, NUMBER, REDIF2, REDSAV, and WRTBM.

E.2.2 INTERFACES

Subprogram SETUP2 uses the named common blocks CLASS GLOBAL, and INFORM. The variable, APRFLG, in common block CLASS will be involved in the modifications to SETUP2.

E.2.3 INPUTS

- Calling sequence:

CALL SETUP2 (ARRAY, TOP, FLDFLG, APRIOR, BMATRX, KATNO)

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
ARRAY	10, 600	In	Variably dimensioned working storage.
TOP	1	In	Maximum usable location in ARRAY - set = 10, 600
FLDFLG	1	In	Unused in SETUP2
APRIOR	60	Out	Storage for the subclass apriori probability values.

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
BMATRX	(BMCOMB, BMFEAT) BMCOMB = no. of linear com- binations in "B"-matrix BMFEAT = no. of channels used in the "B"-matrix.	OUT	Storage for the "B"- transformation matrix
KATNO	60	Out	Class-category correspondence.

E.2.4 OUTPUTS

There will be no change in either file output or line printer output as a result of these modifications to SETUP2.

E.2.5 STORAGE REQUIREMENTS

TBD

E.2.6 DESCRIPTION

The modifications to SETUP2 will include:

1. A test on the flag, APRFLG = -777777, to trigger a transfer to the coding for computation of subclass apriori probability values using subclass population data available in the array, KEPPTS (common block INFORM).
2. The computation and storage of subclass apriori probability values, in the following manner:

Total no. of subclass pixels

$$TKEPTS = \sum_{I=1}^{NOSUB2} KEPPTS(I)$$

apriori probability value for subclass I

$$APRIOR(I) = \frac{KEPPTS(I)}{TKEPTS}$$

I = 1, NOSUB2 (NOSUB2 = total no. of subclasses)

E.2.7 FLOW CHART

N/A

E.2.8 LISTING

TBD

APPENDIX F

APPENDIX F

TABLE OF CONTENTS

- F.1 SOFTWARE FOR SUBPROGRAM NO. 1 (SETUP3)
- F.2 SOFTWARE FOR SUBPROGRAM NO. 2 (REDIF3)
- F.3 SOFTWARE FOR SUBPROGRAM NO. 3 (DSPLY2)
- F.4 SOFTWARE FOR SUBPROGRAM NO. 4 (DOTSUM)

F.1 SOFTWARE COMPONENT NO. 1 (SETUP3)

The modifications to SETUP3 will:

1. Implement the input of dot data parameters from the DOTFIL file created by the DOTDATA processor
2. Utilize existing storage in ARRAY normally occupied by training field information to store the dot data parameters
3. Perform a check for agreement between the set of dot category names and the set of classification category names, and
4. Order the dot data in storage by ascending line number with the group of dots on a given line ordered by ascending sample number.

The dot performance flag, DOTKEY, will be tested. If DOTKEY=1 activities related to dot performance summary processing will be initiated.

If dot data performance summaries are requested and if there is any mismatch of category names between analyst labeled dot categories (DOTFIL) and classification categories (MAPTAP), dot data processing will be terminated, a message written to warn the user, and the usual DISPLAY processor functions performed as detailed by the input control cards.

F.1.1 LINKAGES

Subprogram SETUP3 is called by the DISPLAY processor driver program, DSPLAY. Subprogram SETUP3 makes calls to the following subprograms: FIND, NUMBER, REDIF3, WRTFLD, and a new subprogram TAPLAB.

Subprogram SETUP3 references two common blocks DISPL and GLOBAL.

F.1.3 INPUTS

- Calling sequence: CALL SETUP3 (ARRAY, TOP)

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
ARRAY	10 600	In	Variably dimensioned working array.
TOP	1	In	Maximum usable location in ARRAY. Set=10 600 in the EOD-LARSYS executive program, MONITOR.

F.1.4 OUTPUTS

Modifications to SETUP3 will result in two additions to line printer output:

1. In the user input summary of requests the following message will be added when the new DOTFILE control card is input:
"PRINT DOT DATA PERFORMANCE SUMMARIES FOR DOT DATA FILE NO. XX FROM TAPE (OR FILE) CCCCCC"
2. If each category name from the input dot data file, DOTFIL, does not agree with one of the category names from the input classification results file, MAPTAP, the following message will be printed:
"DOT DATA PERFORMANCE SUMMARIES WILL NOT BE PRODUCED - THE CATEGORY NAMES FROM MAPTAP AND DOTFILE DO NOT MATCH -
CATEGORY NAMES FROM MAPTAP ARE: MCAT1, MCAT2, MCAT3, ...
CATEGORY NAMES FROM DOTFILE ARE: DCAT1, DCAT2, DCAT3, ..."

F.1.5 STORAGE REQUIREMENTS

TBD

F.1.6 DESCRIPTION

The modifications to subprogram SETUP3 will be as follows:

1. The storage allocation in ARRAY for training fields will be used for storing the dot data information input by the subprogram RDDOTS. Treating each dot as a one pixel training field, recompute the base addresses, FLDSV2, VERTX2, and TOTVT2, for three areas in ARRAY.
2. The tests on TRNKEY will be modified to include a test on the flag, DOTKEY (dot data processing flag). If applicable, initiate the output of a correct list of user-requested options.
3. The dot data performance summary flag, DOTKEY=1, will be used to initiate a comparison of category names input from the classification results file, MAPTAP, against the category name read from the dot data file, DOTFIL. If there is a one-for-one match on category names, dot data performance summaries will be produced by DISPLAY. If there is any mismatch between the two sets of category names, the message described in section F.1.4 (2) above will be written, the DOTKEY flag reset=0, and dot data processing will not be done by the DISPLAY processor.
4. The training field areas will be initialized with dot data information as follows:

a. Storage will be allocated such that

TRNSAV(1,1) = ARRAY(FLDSV2),
TRNVER(1,1) = ARRAY(VERTX2),
TRNFLD(1,1) = ARRAY(FIELD2)

Base addresses for ARRAY storage are computed as

FLDSV2 = 1
VERTX2 = FLDSV2 + 4*NOFLD2,
FIELD2 = VERTX2 + 2*TOTVT2

where total number of fields = NOFLD2 = total number of dots; total number of vertices = TOTVT2 = total number of dots.

- b. Dot data will be stored in the training field areas in the following manner:

TRNSAV(1,I) = unused in SETUP3
TRNSAV(2,I) = category number for dot I
TRNSAV(3,I) = dot type
TRNSAV(4,I) = 1 = number of vertices
TRNVER(1,I) = dot sample number
TRNVER(2,I) = dot line number

For $I = 1, 2, 3, \dots$, NOFLD2 = total number of dots.

Internal subprogram FIXFLD will initialize TRNFLD, to contain the rectangular area surrounding a dot, such that

TRNFLD(1,I) = starting line number
TRNFLD(2,I) = ending line number
TRNFLD(3,I) = starting sample number
TRNFLD(4,I) = ending sample number
TRNFLD(5,I) = pointer to the location in TRNVER for the vertex of the field (dot)

5. The size of the performance table will be set = NOFLD2 = total number of dots (i.e., PCTSZ = NOFLD2 if DOTKEY > 0)
6. After the ordering of the dot data, initialization of TRNSAV, and initialization of TRNFLD, the usual call to the subprogram SETKEY (if DOTKEY > 0), will be bypassed, and SETUP3 will set FLDKEY = 0, DOTKEY = number of dot categories, and will take the normal return to DISPLAY.

F.1.7 FLOW CHART

N/A

F.1.8 LISTING

TBD

F.2 SOFTWARE COMPONENT NO. 2 (REDIF3)

The modifications to subprogram REDIF3 will result in the following additional functions being performed:

1. A new control card DOTFILE will be read and decoded
2. A new flag DOTKEY will be set and used in other subprograms, when dot data classification performance summaries are requested. DOTKEY will be placed in the common block DISPL.

F.2.1 LINKAGES

Subprogram REDIF3 is called by subprogram SETUP3. Subprogram REDIF3 calls the following subprograms: CHIN, FIND, FLTNUM, LAREAD, NXTCHR, and a new subprogram RDDOTS.

F.2.2 INTERFACES

Subprogram REDIF3 references the common blocks DISPL and GLOBAL. DOTKEY will be added to the common block DISPL, tested in subprograms SETUP3, DSPLY2, and DOTSUM, and used for initiating dot data processing.

F.2.3 INPUTS

The calling sequence for subprogram REDIF3 will be unchanged by these modifications.

- Calling sequence: CALL REDIF3(TSTSAV,TSTFLD,TSTVER,VDIM)

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
TSTSAV	(4,250)	Out	Used for test field or designated field information: TSTSAV(1,N) = field name TSTSAV(2,N) = class number TSTSAV(3,N) = subclass TSTSAV(4,N) = number of vertices for field N

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
			N = 1,2,3,...,number of fields.
TSTFLD	(5,200)	Out	Used for storage of coordinates of the rectangular area surrounding a test or designated field.
TSTVER	VDIM	Out	Used for storage of the sample, line coordinates of the vertices for all input test or designated fields.
VDIM	1	In	Total number of locations allocated for storage of all test or designated field vertices in the variably-dimensioned ARRAY in SETUP3.

● New control card

<u>Keyword</u>	<u>Parameter(s)</u>	<u>Function</u>
DOTFILE	UNIT=m, FILE=n (Default: UNIT=8, FILE=1)	Implements the reading of dot data from the specified (or default) unit and file in SETUP3 and the output of dot performance summaries in DSPLY2.

F.2.4 OUTPUTS

The line printer output will list the new control card at the time it is read by REDIF3

F.2.5 STORAGE REQUIREMENTS

TBD

F.2.6 DESCRIPTION

The order of the modifications to subprogram REDIF3 are:

1. The keyword array, OPT, will be redimensioned to 13 locations, and the new control card keyword, DOTFIL, will be added to OPT.
2. The new dot data processing flag, DOTKEY = 0, will be initialized.
3. For match-up of keyword DOTFIL in OPT array versus input control card keyword, a transfer to a new statement number 785 will be added to the transfer list.
4. Statement number 785 and coding to read the parameters field from the DOTFILE control card, coding to set the new flag, DOTKEY = 1, and the existing flag, TRNKEY = 1 will be added.
5. After existing statement number 870, if DOTKEY > 0, a test for input of designated fields will be performed. The fields may be either "DESIGNATED OTHER" or "DESIGNATED UNIDENTIFIABLE." For dot data processing, the test field flag, TSTKEY, will be turned off (TSTKEY=0). The designated field flag DESKEY will be turned on (DESKEY=1), if and only if any fields read in are identified as "DESIGNATED" fields.

F.2.7 FLOW CHART

N/A

F.2.8 LISTING

TBD

F.3 SOFTWARE COMPONENT NO. 3 (DSPLY2)

DSPLY2 will be modified to test the dot performance flag, DOTKEY. If DOTKEY > 0 a call to a new subprogram DOTSUM will be initiated. DOTSUM will produce the two dot data classification summaries. The capability to border each dot on the classification line printer map, as is currently done with training fields, will be preserved.

F.3.1 LINKAGES

Subprogram DSPLY is called by the DISPLAY processor driver program DSPLAY. Subprogram DSPLY2 calls the following subprograms: DESIG, FLDBOR, MAPHD, PCT, PRTSUM, WRTHEd, WRTLIN, and a new subprogram, DOTSUM.

F.3.2 INTERFACES

Subprogram DSPLY2 references two common blocks DISPL and GLOBAL.

F.3.3 INPUTS

The calling sequence for DSPLY2 will be unchanged as a result of these modifications. The usage of the three training field information arrays in the calling arguments TRNSAV, TRNFLD, and TRNVER will be revised by these modifications. Also, the usage of the classification performance table PCTAB, is modified for dot data performance tabulation.

- Calling sequence: CALL DSPLY2(TRNSAV,TRNFLD,TRNVER,TSTSAV,
TSTFLD,TSTVER,PCTAB)

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
TRNSAV	(4,NOFLD2) NOFLD2 = total number of dots	In	Initialized in SETUP3 as follows: TRNSAV(1,I) = unused TRNSAV(2,I) = unused TRNSAV(3,I) = dot type

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
			TRNSAV(4,I) = 1 = number of vertices for I = 1,2,...,NOFLD2 = total number of dots
TRNFLD	(5,NOFLD2)	In	Coordinates of rectangular area surrounding each dot. Initialized in SETUP3 as follows: TRNFLD(1,I) = starting line number TRNFLD(2,I) = ending line number TRNFLD(3,I) = starting sample number TRNFLD(4,I) = ending sample number TRNFLD(5,I) = location in TRNVER for vertex of dot I. I = 1,2,...,NOFLD2 = total number of dots
TRNVER	(2,TOTVT2) TOTVT2 = total number of vertices for all dots = total number of dots	In	Array storage for training field (dot) vertex coordinates: TRNVER(1,I) = sample number TRNVER(2,I) = line number I = 1,2,..., total number of dots = NOFLD2

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
TSTSAV	(4,NOFLD3) NOFLD3 = total number of test designated fields	In	Array storage for test or designated fields, equivalent to TRNSAV for training fields.
TSTFLD	(5,NOFLD3)	In	Used for test or design- ated fields, equivalent to TRNFLD for training fields.
TSTVER	(2,TOTVT3) TOTVT3 = total number of ver- tices for all test or design- ated fields	In	Used for test or design- ated fields, information equivalent to TRNVER for training fields.
PCTAB	(NOFLD2,1) NOFLD2 = total number of dots		For dot performance tabulation, PCTAB(I,1) = classified subclass for dot I, I = 1,2,...,NOFLD2.

F.3.4 OUTPUTS

There is no change in output from DSPLY2 as a result of the modifications.

F.3.5 STORAGE REQUIREMENTS

TBD

F.3.6 DESCRIPTION

Two modifications will be implemented in DSPLY2. Following the usual printout of the classification summary for the classified field, if DOTKEY > 0, PCTAB(1,1) will be set = -7654321 to be used

as a flag in subprogram PCT, and a call will be made to the new subprogram, DOTSUM. The dot classification performance table, PCTAB, obtained from subprogram PCT will be passed to the new subprogram, DOTSUM, along with classification totals over the total area classified. On return from DOTSUM, DSPLY2 will continue its normal flow back to reading of the MAPTAP for the next field classified (if any).

F.3.7 FLOW CHART

N/A

F.3.8 LISTING

TBD

F.4 SOFTWARE COMPONENT NO. 4 (DOTSUM)

Subprogram DOTSUM is a new subprogram to be added to the DISPLAY processor. The functions to be performed by the subprogram are as follows:

1. To compute proportions over the classified area for the following:
 - a. Uncorrected dot categories
 - b. Bias corrected dot categories
2. To compute the alpha table
3. To output two performance tables as follows
 - a. Summary by dot category showing:
 - 1.) Total number of dots labeled by analyst
 - 2.) Percent of correct classification of analyst-labeled dots
 - 3.) Number of dots thresholded
 - 4.) Uncorrected category proportion
 - 5.) Bias corrected category proportion
 - 6.) Alpha values
 - 7.) Number of dots in category classified into other categories
 - b. Summary by individual dot showing:
 - 1.) Coordinate of dot (sample, line)
 - 2.) Analyst label
 - 3.) Classifier label (category and subclass)

F.4.1 LINKAGES

Subprogram DOTSUM is called by subprogram DSPLY2. The subprogram does not reference any other subprograms.

F.4.2 INTERFACES

Common block DISPL will be utilized by subprogram DOTSUM.

F.4.3 INPUTS

- Calling sequence: CALL DOTSUM(SUBTOT, THTOT, PCTAB, TRNSAV, TRNVER, SCRACH)

<u>Parameter</u>	<u>Dimension</u>	<u>In/out</u>	<u>Definition</u>
SUBTOT	NOSUB3 (number of subclasses +1)	In	Total number of pixels classified into sub- classes and thresholded
THTOT	NOSUB3	In	Total number of pixels thresholded in each sub- class, over the total area classified
PCTAB	(NOFLD2,1) NOFLD2 = total number of dots	In	For DOT performance, PCTAB(I,1) = subclass for dot I
TRNSAV	(4,NOFLD2)	In	TRNSAV(1,N) = unused TRNSAV(2,N) = labeled category number TRNSAV(3,N) = dot type TRNSAV(4,N) = 1 (number of vertices)
TRNVER	2, TOTVT2	In	Sample, line dot coordinates
SCRACH	3000 (=IR array, IN DSPLY2)	In	Scratch storage for working arrays in DOTSUM

F.4.4 OUTPUTS

The dot data classification summaries are provided in place of test or training field summaries.

- Dot category summary table (see table F-1).
- Individual dot summary table (see table F-2).

F.4.5 STORAGE REQUIREMENTS

TBD

F.4.6 DESCRIPTION

The flow of the subprogram will be as follows:

1. Input (via calling arguments) and common block DISPL:
 - a. SUBTOT (NOSUB3) = total number of pixels classified into subclass 1,2,...,NOSUB2,THRESH
(NOSUB2) = total number of subclasses
(NOSUB3) = total number of subclasses + 1
 - b. THTOT (NOSUB3) = total pixels thresholded, by subclass
 - c. PCTAB (NOFLD2,NOSUB3) = total pixels classified into each subclass and thresholded, for each dot i, i = 1,2,...,NOFLD2 = total number of dots
 - d. TRNSAV (4,NOFLD2):
TRNSAV(1,N) = unused
TRNSAV(2,N) = dot category number from the MAPTAP category matching the dot labeled category
TRNSAV(3,N) = dot type
TRNSAV(4,N) = 1 = number of vertices for dot
 - e. SCRACH(3000) = scratch storage to use as needed internally in DOTSUM

Figure F-1.- DOT DATA PERFORMANCE SUMMARY

[Summary of labeled versus classified categories]

Analyst labeled dot category	Total number analyst labeled dots	Percent correctly classified	Number thresholded	Uncorrected category proportion	Bias corrected proportion	Classification (alpha)		
						C ₁	C ₂	C ₃
C ₁	1 000	85	0	14.3	16.9	850 (99)	125 (6)	25 (0.9)
C ₂	2 000	97	0	37.8	33.4	10 (1.2)	1 940 (85.6)	50 (1.8)
C ₃	3 000	93	0	47.9	49.9	0 (0.0)	200 (8.8)	2 800 (97)

TABLE F-2.- DOT DATA PERFORMANCE SUMMARY

[Summary of individual dots]
DOT ID = (sample, line)

(1,1)		(2,1)		(3,1)		...	
<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>		
WHEAT	WHEAT (WHT1)	OTHER	OTHER (OTH2)	NONWHT	OTHER (OTH3)		
(1,2)		(4,2)		(7,2)		...	
<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>		
NONWHT	NONWHT (NWH2)	WHEAT	WHEAT (WHT3)	OTHER	NONWHT (NWH4)		
(1,N)		(6,N)		(7,N)		...	
<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>	<u>Label</u>	<u>Category classified (subclasses)</u>		
OTHER	OTHER (OTH3)	OTHER	NONWHT (NWH3)	NONWHT	NONWHT (NWH1)		

::

- f. From common block DISPL: NOCAT = number of classification categories; CATNAM(61) = classification category names; SUBCAT(60) = subclass-to-category correspondence (SUBCAT(I) = M, means subclass number I belongs in category number M); NOFLD2 = total number of dots; DOTKEY = number of dot categories
2. Store dot type in TYPE (=TRNSAV(3,1)), initialize NDCAT = DOTKEY = number of dot categories and initialize NCLAT = NOCAT = number of classification categories.
 3. Move labeled category from TRNSAV(2,I) to TRNSAV(3,I), for I = 1, NOFLD2 (=total number of dots)
 4. Transfer each dot "vertex" (sample, line coordinate) from TRNVER array to TRNSAV (1,I), TRNSAV (2,I) for I = 1, NOFLD2 dots TRNSAV(1,I) = dot sample, TRNSAV(2,I) = dot line number, TRNSAV(3,I) = labeled category number, I = 1, NOFLD2 = total number of dots.
 5. TRNVER and remainder of ARRAY, except PCTAB, now available for scratch storage as needed in DOTSUM.
 6. Compute total number of pixels classified in the total area classified, except for thresholded pixels

$$\text{PIXTOT} = \sum_{i=1}^{\text{NOSUB2}} \text{SUBTOT}(i)$$
 7. Compute dot totals classified, by category. Initialize CLTOT(K) = 0, K = 1,2,...,NCLCAT(=NOCAT). Then, for I = 1, NOFLD2, SUBCL = (PCTAB(I,1), 1 ≤ SUBCL ≤ NOSUB2, CAT = SUBCAT(SUBCL), CLTOT(CAT) = CLTOT(CAT) + 1
 8. Based on occurrence in the dot data array, TRNSAV, initialize an ordered set of dot category labels [DOTLBL(J,1) = CATNAM(LBL) = category name and DOTLBL(J,2) = LBL = category number of labeled dot.

[for LBL = TRNSAV(3,I), I = 1, NOFLD2, and DOTLBL(J,2) \neq LBL,
and for J = 1, NDCAT = number of dot categories]

9. Compute the uncorrected proportions of dot categories over the total area classified; $PUNC(I) = CLTOT(J)/PIXTOT*100$; I = 1, NDCAT; J = DOTLBL(I,2).
10. Initialize TRNSAV(4,I), I = 1, NOFLD2 (number of dots) to contain the classification category (number) for dot I; SUBCL = PCTAB(I,J), J = 1, NOSUB3 = number of subclasses including thresholded, if SUBCL = NOSUB3, CAT = NOCAT + 1 if SUBCL \neq NOSUB3, CAT = SUBCAT(SUBCL). TRNSAV(4,I) = CAT (classified category number).

Set NCLCAT = NOCAT + 1 (i.e., number of classification categories, plus one to account for the thresholded dots).
11. Initialize LABCLS(I,J) = 0, I = 1,2,...,NDCAT; J = 1,2,..., NCLCAT (=NOCAT + 1)
12. Form a table showing (a) the labeled dot category, and (b) the total number of dots with a given label that have been classified into each classification category.

Initialize LABCLS(I,J) = 0, I = 1, NDCAT, J = 1, NCLCAT; then for I = 1, NDCAT; DLBL = DOTLBL (I,2) and for J = 1, NOFLD2 where TRNSAV (3,J) = DLBL, then CLS = TRNSAV (4,J) and LABCLS (I,CLS) = LABCLS (I,CLS) + 1.
13. Compute total number of dots classified into each category; initialize LBLTOT = 0, then for I = 1, NDCAT and J = 1, NCLCAT LBLTOT(I) = LBLTOT(I) + LABCLS(I,J).
14. Compute alpha table for labeled versus classified dots; $\alpha(I,J) = LBLCLS (J,I)/LBLTOT(I)*100$, I = 1, NDCAT classification categories, J = 1, NDCAT labeled categories, set $\alpha(I,J) = 0$, for I = N where N > NDCAT.
15. Compute corrected proportions for labeled dot categories; for I = 1, NDCAT:
labeled categories

$$PCORR(I) = \sum_{J=1}^{NDCAT} [\alpha(J,1) * PUNC(J)] * 100$$

16. Tabulate the total number of dots in each dot category; initialize TOTLBL = 0, then for I = 1, NOFLD2 = total number of dots and LBL = TRNSAV(3,I) and for J where DOTLBL(J,2) = LBL; TOTLBL(J) = TOTLBL(J) + 1.
17. Tabulate the number of dots in each category which were thresholded; initialize TOTTHR(J) = 0, J = 1, NDCAT; then for I = 1, NOFLD2 = total number of dots; when TLBL = TRNSAV(4,I) = NDCAT(=NOCAT + 1) and when, for LBL = TRNSAV(3,I) and DOTLBL(J,2) = LBL, J = 1, NDCAT, then TOTTHR(J) = TOTTHR(J) + 1.
18. Print first dot classification performance summary, by category; for I = 1, NDCAT
 - a. labeled dot category name = DOTLBL(I,1)
 - b. total number of analyst-labeled dots in each category = LBLTOT(I)
 - c. percent correct classification = CLTOT(I)/LBTOT(I)*100
 - d. number of dots thresholded in the analyst-labeled category = TOTTHR(I)
 - e. uncorrected labeled category proportion = PUNC(I)
 - f. bias corrected labeled category proportion = PCORR(I)
 - g. classification summary and α_{JI} for category I; for J = 1, NCLCAT, LABCLS(I,J), ALPHA(J,I)
19. Set SUBNAM(NOSUB3) = 'THRESH'
20. Print second dot classification performance summary, by individual dot; for I = 1, NOFLD2 = total number of dots.
 - a. dot identification = (sample,line) = TRNSAV(1,I), TRNSAV(2,I)

- b. labeled category = CATNAM[TRNSAV(3,I)]
- c. classified category = CATNAM[TRNSAV(4,I)]
- d. classified subclass = SUBNAM[PCTAB(I,1)]

21. Return to calling subprogram, DSPLY2.

F.4.7 FLOW CHART

See section F.4.6 for a narrative flow chart of the subprogram.

F.4.8 LISTING

TBD

APPENDIX G

DOT DATA FILE FORMAT
(DOTFIL)

The tape, DOTFIL is output by the DOTDATA processor. The records are written with an unformatted FORTRAN write.

A file is output for each TYPE of field(s). The file consists of the following records:

Repeat for each TYPE	{	Rec. No. 1	}	field information
		Rec. No. 2		
	{	Rec. No. 3	}	data record
		E-O-F		

TYPE 1 consists of labeling dots; TYPE 2 consists of bias correction dots.

Rec. No. 1

WRITE (DOTUNT) NOCAT, NOFEAT, NOFLD, TOTVRT, TOTDOT, NOSUN,
(CATNAM(I), I=1, NOCAT), SIZE

<u>Parameter</u>	<u>Dimension</u>	<u>Definition</u>
NOCAT	1	Number of category names.
NOFEAT	1	Number of channels.
NOFLD	1	Number of fields.
TOTVRT	1	Number of vertices.
TOTDOT	1	Number of dots
NOSUN	1	Number of sun angles.
CATNAM	NOCAT	Array containing the category names.
SIZE	1	4 + NOFEAT

Rec. No. 2

```
WRITE (DOTUNT) (FETVEC(I), I=1, NOFEAT)
               ((FLDSAV(I,J), I=1,4), J=1, NOFLD),
               ((VERTEX(I,J), I=1,2), J=1, TOTVRT),
               (ANGLE(I), I=1, NOSUN)
```

<u>Parameter</u>	<u>Dimension</u>	<u>Definition</u>
FETVEC	NOFEAT	Array containing the channel numbers.
FLDSAV	(4,NOFLD)	Array containing the field description.
VERTEX	(2,NOFLD)	Array containing the field vertices.
ANGLE	NOSUN	Array containing the sun angles.

Rec. No. 3

```
WRITE (DOTUNT) ((DOTS(I,J), I=1, SIZE), J=1, TOTDOT)
```

<u>Parameter</u>	<u>Dimension</u>	<u>Definition</u>
DOTS	(TOTDOT, SIZE)	Array containing the dot information. DOTS(1,I)=sample number for dot i. DOTS(2,I)=line number for dot i. DOTS(3,I)=type number for dot i. DOTS(4,I)=category number for dots i (optional)

Parameter

Dimension

Definition

DOTS(5,I)

.

.

.

=dot

vector i

DOTS(4+NOFEAT,I)